Probabilistic Robust Timed Games

Youssouf Oualhadj, Pierre-Alain Reynier, Ocan Sankur

Probabilistic Robust Timed Games *

Youssouf Oualhadj[1], Pierre-Alain Reynier[2], and Ocan Sankur[3]

[1] Université de Mons (UMONS), Belgium
[2] LIF, Université d'Aix-Marseille & CNRS, France
[3] Université Libre de Bruxelles, Belgium

Abstract. Solving games played on timed automata is a well-known problem and has led to tools and industrial case studies. In these games, the first player (Controller) chooses delays and actions and the second player (Perturbator) resolves the non-determinism of actions. However, the model of timed automata suffers from mathematical idealizations such as infinite precision of clocks and instantaneous synchronization of actions. To address this issue, we extend the theory of timed games in two directions. First, we study the synthesis of *robust* strategies for Controller which should be tolerant to adversarially chosen clock imprecisions. Second, we address the case of a stochastic perturbation model where both clock imprecisions and the non-determinism are resolved randomly. These notions of robustness guarantee the implementability of synthesized controllers. We provide characterizations of the resulting games for Büchi conditions, and prove the EXPTIME-completeness of the corresponding decision problems.

1 Introduction

For real-time systems, timed games are a standard mathematical formalism which can model control synthesis problems under timing constraints. These consist in two-players games played on arenas, defined by timed automata, whose state space consists in discrete locations and continuous clock values. The two players represent the control law and the environment. Since the first theoretical works [2], symbolic algorithms have been studied [10], tools have been developed and successfully applied to several case studies.

Robustness Because model-based techniques rely on abstract mathematical models, an important question is whether systems synthesized in a formalism are implementable in practice. In timed automata, the abstract mathematical semantics offers arbitrarily precise clocks and time delays, while real-world digital systems have response times that may not be negligible, and control software cannot ensure timing constraints exactly, but only up to some error, caused by clock imprecisions, measurement errors, and communication delays. A major challenge is thus to ensure that the synthesized control software is *robust*, *i.e.* ensures the specification even in presence of imprecisions [15].

Following these observations there has been a growing interest in lifting the theory of verification and synthesis to take robustness into account. Model-checking problems were re-visited by considering an unknown perturbation

* This work was partly supported by ANR projects ECSPER (ANR-2009-JCJC-0069) and ImpRo (ANR-2010-BLAN-0317), European project Cassting (FP7-ICT-601148), and the ERC starting grant inVEST (FP7-279499).

parameter to be synthesized for several kinds of properties [18, 12, 7], see also [9]. Robustness is also a critical issue in controller synthesis problems. In fact, due to the infinite precision of the semantics, synthesized strategies may not be realizable in a finite-precision environment; the controlled systems synthesized using timed games technology may not satisfy the proven properties at all. In particular, due to perturbations in timings, some infinite behaviors may disappear completely. A first goal of our work is to develop algorithms for *robust* controller synthesis: we consider this problem by studying robust strategies in timed games, namely, those guaranteeing winning despite imprecisions bounded by a parameter.

Adversarial or Stochastic Environments We consider controller synthesis problems under two types of environments. In order to synthesize correct controllers for critical systems, one often considers an *adversarial* (or worst-case) environment, so as to ensure that *all* behaviors of the system are correct. However, in some cases, one is rather interested considering a *stochastic environment* which determines the resolution of non-determinism, and the choice of clock perturbations following probability distributions. We are then interested in satisfying a property *almost-surely*, that is, with probability 1, or *limit-surely*, that is, for every $\varepsilon > 0$, there should exist a strategy for Controller under which the property is satisfied with probability at least $1 - \varepsilon$.

Contributions We formalize the robust controller synthesis problem against an adversarial environment as a (non-stochastic) game played on timed automata with an unknown imprecision parameter δ, between players Controller and Perturbator. The game proceeds by Controller suggesting an action and a delay, and Perturbator perturbing each delay by at most δ and resolving the non-determinism by choosing an edge with the given action. Thus, the environment's behaviors model both uncontrollable moves and the limited precision Controller has. We prove the EXPTIME-completeness of deciding whether there exists a positive δ for which Controller has a winning strategy for a Büchi objective, matching the complexity of timed games in the classical sense. Our algorithm also allows one to compute $\delta > 0$ and a witness strategy on positive instances.

For stochastic environments, we study two probabilistic variants of the semantics: we first consider the case of adversarially resolved non-determinism and independently and randomly chosen perturbations, and then the case where both the non-determinism and perturbations are randomly resolved and chosen. In each case, we are interested in the existence of $\delta > 0$ such that Controller wins almost-surely (resp. limit-surely). We give decidable characterizations based on finite abstractions, and EXPTIME algorithms. All problems are formulated in a parametric setting: the parameter δ is unknown and is to be computed by our algorithms. This is one of the technical challenges in this paper.

Our results on stochastic perturbations can also be seen as a new interpretation of robustness phenomena in timed automata. In fact, in the literature on robustness in timed automata, non-robust behaviors are due to the accumulation of the imprecisions δ along long runs, and in the proofs, one exhibits non-robustness by artificially constructing such runs (e.g. [12, 21]). In contrast, in the present setting, we show that non-robust behaviors either occur almost surely, or can be avoided surely (Theorem 13).

Related Work While several works have studied robustness issues for model-checking, there are very few works on robust controller synthesis in timed systems:

– The (non-stochastic) semantics we consider was studied for *fixed* δ in [11] encoding by timed games; but the parameterized version of the problem was not considered.
– The restriction of the parameterized problem to (non-stochastic) *deterministic* timed automata was considered in [21]. Here, the power of Perturbator is restricted as it only modifies the delays suggested by Controller, but has no non-determinism to resolve. Therefore, the results consist in a robust Büchi acceptance condition for timed automata, but they do not generalize to timed games. Technically, the algorithm consists in finding an *aperiodic* cycle, which are cycles that are "stable" against perturbations. This notion was defined in [3] to study entropy in timed languages. We will also use aperiodic cycles in the present paper.
– A variant of the semantics we consider was studied in [8] for (deterministic) timed automata and shown to be EXPTIME-complete already for reachability due to an implicit presence of alternation. Timed games, Büchi conditions, or stochastic environments were not considered.
– Probabilistic timed automata where the non-determinism is resolved following probability distributions have been studied [16, 4, 17]. Our results consist in deciding almost-sure and limit-sure Büchi objectives in PTAs subject to random perturbations in the delays. Note that PTAs are equipped with a possibly different probability distribution for each action. Although we only consider uniform distributions, the two settings are equivalent for almost-sure and limit-sure objectives. Games played on PTA were considered in [14] for minimizing expected time to reachability with NEXPTIME $\cap$ co-NEXPTIME algorithms.

To the best of our knowledge, this work is the first to study a stochastic model of perturbations for synthesis in timed automata.

Organization Definitions are given in Section 2. Preliminaries concerning the notions of regions graphs, orbit graphs and shrunk difference bound matrices are presented in Section 3. The finite-state game abstraction used to characterize the positive instances of the (non-stochastic) problem is presented in Section 4 and the proof of correction is given in Section 5. In Section 6, we consider the two stochastic models for the environment.

2 Robust Timed Games

Timed automata. Given a finite set of clocks $\mathcal{C}$, we call *valuations* the elements of $\mathbb{R}_{\geq 0}^{\mathcal{C}}$. For a subset $R \subseteq \mathcal{C}$ and a valuation ν, $\nu[R \leftarrow 0]$ is the valuation defined by $\nu[R \leftarrow 0](x) = 0$ if $x \in R$ and $\nu[R \leftarrow 0](x) = \nu(x)$ otherwise. Given $d \in \mathbb{R}_{\geq 0}$ and a valuation ν, the valuation $\nu + d$ is defined by $(\nu + d)(x) = \nu(x) + d$ for all $x \in \mathcal{C}$. We extend these operations to sets of valuations in the obvious way. We write $\mathbf{0}$ for the valuation that assigns 0 to every clock.

An atomic clock constraint is a formula of the form $k \preceq x \preceq' l$ or $k \preceq x - y \preceq' l$ where $x, y \in \mathcal{C}$, $k, l \in \mathbb{Z} \cup \{-\infty, \infty\}$ and $\preceq, \preceq' \in \{<, \leq\}$. A *guard* is a conjunction of atomic clock constraints. A valuation ν satisfies a guard g, denoted $\nu \models g$, if all constraints are satisfied when each $x \in \mathcal{C}$ is replaced with $\nu(x)$. We write $\Phi_{\mathcal{C}}$ for the set of guards built on $\mathcal{C}$. A *zone* is a subset of $\mathbb{R}_{\geq 0}^{\mathcal{C}}$ defined by a guard.

A *timed automaton* $\mathcal{A}$ over a finite alphabet of actions Act is a tuple $(\mathcal{L}, \mathcal{C}, \ell_0, \mathsf{Act}, E)$, where $\mathcal{L}$ is a finite set of locations, $\mathcal{C}$ is a finite set of clocks, $E \subseteq \mathcal{L} \times \Phi_{\mathcal{C}} \times \mathsf{Act} \times 2^{\mathcal{C}} \times \mathcal{L}$ is a set of edges, and $\ell_0 \in \mathcal{L}$ is the initial location. An edge

$e = (\ell, g, a, R, \ell')$ is also written as $\ell \xrightarrow{g,a,R} \ell'$. A state is a pair $q = (\ell, \nu) \in \mathcal{L} \times \mathbb{R}^{\mathcal{C}}_{\geq 0}$. An edge $e = (\ell, g, a, R, \ell')$ is enabled in a state (ℓ, ν) if ν satisfies the guard g.

The set of possible behaviors of a timed automaton can be described by the set of its runs, as follows. A *run* of $\mathcal{A}$ is a sequence $q_1 e_1 q_2 e_2 \dots$ where $q_i \in \mathcal{L} \times \mathbb{R}^{\mathcal{C}}_{\geq 0}$, and writing $q_i = (\ell, \nu)$, either $e_i \in \mathbb{R}_{>0}$, in which case $q_{i+1} = (\ell, \nu + e_i)$, or $e_i = (\ell, g, a, R, \ell') \in E$, in which case $\nu \models g$ and $q_{i+1} = (\ell', \nu[R \leftarrow 0])$. The set of runs of $\mathcal{A}$ starting in q is denoted $\mathsf{Runs}(\mathcal{A}, q)$.

Parameterized timed game. In order to define perturbations, and to capture the reactivity of a controller to these, we define the following parameterized timed game semantics. Intuitively, the parameterized timed game semantics of a timed automaton is a two-player game parameterized by $\delta > 0$, where Player 1, also called Controller chooses a delay $d > \delta$ and an action $a \in \mathsf{Act}$ such that every a-labeled enabled edge is such that its guard is satisfied after any delay in the set $d + [-\delta, \delta]$ (and there exists at least one such edge). Then, Player 2, also called Perturbator chooses an actual delay $d' \in d + [-\delta, \delta]$ after which the edge is taken, and chooses one of the enabled a-labeled edges. Hence, Controller is required to always suggest delays that satisfy the guards whatever the perturbations are.

Formally, given a timed automaton $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \ell_0, \mathsf{Act}, E)$ and $\delta > 0$, we define the *parameterized timed game* of $\mathcal{A}$ w.r.t. δ as a two-player turn-based game $\mathcal{G}_\delta(\mathcal{A})$ between players Controller and Perturbator. The state space of $\mathcal{G}_\delta(\mathcal{A})$ is partitioned into $V_C \cup V_P$ where $V_C = \mathcal{L} \times \mathbb{R}^{\mathcal{C}}_{\geq 0}$ belong to Controller, and $V_P = \mathcal{L} \times \mathbb{R}^{\mathcal{C}}_{\geq 0} \times \mathbb{R}_{\geq 0} \times \mathsf{Act}$ belong to Perturbator. The initial state is $(\ell_0, \mathbf{0}) \in V_C$. The transitions are defined as follows: from any state $(\ell, \nu) \in V_C$, there is a transition to $(\ell, \nu, d, a) \in V_P$ whenever $d > \delta$, for every edge $e = (\ell, g, a, R, \ell')$ such that $\nu + d \models g$, we have $\nu + d + \varepsilon \models g$ for all $\varepsilon \in [-\delta, \delta]$, and there exists at least one such edge e. Then, from any such state $(\ell, \nu, d, a) \in V_P$, there is a transition to $(\ell', \nu') \in V_C$ iff there exists an edge $e = (\ell, g, a, R, \ell')$ as before, and $\varepsilon \in [-\delta, \delta]$ such that $\nu' = (\nu + d + \varepsilon)[R \leftarrow 0])$. A *play* of $\mathcal{G}_\delta(\mathcal{A})$ is a finite or infinite sequence $q_1 e_1 q_2 e_2 \dots$ of states and transitions of $\mathcal{G}_\delta(\mathcal{A})$, with $q_1 = (\ell_0, \mathbf{0})$, where e_i is a transition from q_i to q_{i+1}. It is said to be *maximal* if it is infinite or cannot be extended. A *strategy* for Controller is a function that assigns to every non-maximal play ending in some $(\ell, \nu) \in V_C$, a pair (d, a) where $d > \delta$ and a is an action such that there is a transition from (ℓ, ν) to (ℓ, ν, d, a). A strategy for Perturbator is a function that assigns, to every play ending in (ℓ, ν, d, a), a state (ℓ', ν') such that there is a transition from the former to the latter state. A play ρ is *compatible* with a strategy f for Controller if for every prefix ρ' of ρ ending in V_C, the next transition along ρ after ρ' is given by f. We define similarly compatibility for Perturbator's strategies. A play naturally gives rise to a unique run, where the states are in V_C, and the delays and the edges are those chosen by Perturbator.

Robust timed game problem. Given $\delta > 0$, and a pair of strategies f, g, respectively for Controller and Perturbator, we denote ρ the unique maximal run that is compatible with both f and g. A *Büchi objective* is a subset of the locations of $\mathcal{A}$. A Controller's strategy f is winning for a Büchi objective B if for any Perturbator's strategy g the run ρ that is compatible with f and g is infinite and visits infinitely often a location of B. The *robust timed game problem* asks, for a timed automaton $\mathcal{A}$ and a Büchi objective B, if there exists $\delta > 0$ such

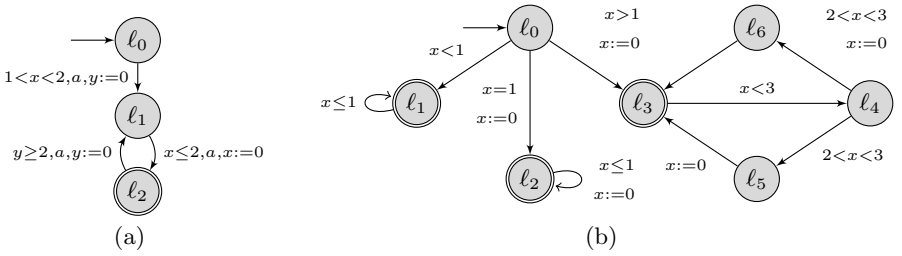

Fig. 1. On the left, a timed automaton from [18] that is not robustly controllable for the Büchi objective $\{\ell_2\}$. In fact, Perturbator can enforce that the value of x be increased by δ at each arrival at ℓ_1, thus blocking the run eventually (see [21]). On the right, a timed automaton that is robustly controllable for the Büchi objective $\{\ell_1, \ell_2, \ell_3\}$. We assume that all transitions have the same label. The cycle around ℓ_1 cannot be taken forever, as value of x increases due to perturbations. The cycle around ℓ_2 can be taken forever, but Controller cannot reach ℓ_2 due to the equality $x = 1$. Controller's strategy is thus to loop forever around ℓ_3. This is possible as for both choices of Perturbator in location ℓ_4, clock x will be reset, and thus perturbations do not accumulate. If one of the two resets were absent, Perturbator could force the run to always take that branch, and would win the game.

that Controller has a winning strategy in $\mathcal{G}_\delta(\mathcal{A})$ for the objective B. When this holds, we say that Controller wins the robust timed game for $\mathcal{A}$, and otherwise that Perturbator does. Note that these games are determined since for each $\delta > 0$, the semantics is a timed game.

Figure 1 shows examples of timed automata where either Controller or Perturbator wins the robust timed game according to our definitions. The main result of this paper for this non-stochastic setting is the following.

Theorem 1. *The robust timed game problem is EXPTIME-complete.*

EXPTIME-hardness is proved in Appendix D. We focus on presenting the EXPTIME membership in Sections 4 and 5. The algorithm relies on a characterization of winning strategies in a refinement of the region automaton construction.

In order to formally introduce the appropriate notions for this characterization, we need definitions given in the following section.

3 Regions, Orbit Graphs and Shrunk DBMs

Regions and Region Automata. Following [12, 18, 3], we assume that the clocks are bounded above by a known constant [4] in all timed automata we consider. Fix a timed automaton $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \ell_0, \mathsf{Act}, E)$. We define *regions* as in [1], as subsets of $\mathbb{R}^{\mathcal{C}}_{\geq 0}$. Any region r is defined by fixing the integer parts of the clocks, and giving a partition $X_0, X_1, \ldots, X_m$ of the clocks, ordered according to their fractional values: for any $\nu \in r$, $0 = \mathsf{frac}(\nu(x_0)) < \mathsf{frac}(\nu(x_1)) < \ldots < \mathsf{frac}(\nu(x_m))$ for any $x_0 \in X_0, \ldots, x_m \in X_m$, and $\mathsf{frac}(\nu(x)) = \mathsf{frac}(\nu(y))$ for any $x, y \in X_i$. Here, $X_i \neq \emptyset$ for all $1 \leq i \leq m$ but X_0 might be empty. For any valuation ν,

[4] Any timed automaton can be transformed to satisfy this property.

let $[\nu]$ denote the region to which ν belongs. $\mathsf{Reg}(\mathcal{A})$ denotes the set of regions of $\mathcal{A}$. A region r is said to be *non-punctual* if it contains some $\nu \in r$ such that $\nu + [-\varepsilon, \varepsilon] \subseteq r$ for some $\varepsilon > 0$. It is said *punctual* otherwise. By extension, we say that (ℓ, r) is non-punctual if r is.

We define the *region automaton* as a finite automaton $\mathcal{R}(\mathcal{A})$ whose states are pairs (ℓ, r) where $\ell \in \mathcal{L}$ and r is a region. Given $(r', a) \in \mathsf{Reg}(\mathcal{A}) \times \mathsf{Act}$, there is a transition $(\ell, r) \xrightarrow{(r',a)} (\ell', s)$ if r' is non-punctual [5], there exist $\nu \in r$, $\nu' \in r'$ and $d > 0$ such that $\nu' = \nu + d$, and there is an edge $e = (\ell, g, R, \ell')$ such that $r' \models g$ and $r'[R \leftarrow 0] = s$. We write the *paths* of the region automaton as $\pi = q_1 e_1 q_2 e_2 \dots q_n$ where each q_i is a state, and $e_i \in \mathsf{Reg}(\mathcal{A}) \times \mathsf{Act}$, such that $q_i \xrightarrow{e_i} q_{i+1}$ for all $1 \leq i \leq n-1$. The *length* of the path is n, and is denoted by $|\pi|$. If a Büchi condition B is given, a cycle of the region automaton is *winning* if it contains an occurrence of a state (ℓ, r) with $\ell \in B$.

Vertices and Orbit Graphs. A *vertex* of a region r is any point of $\bar{r} \cap \mathbb{N}^{\mathcal{C}}$, where $\bar{r}$ denotes the topological closure of r. Let $\mathcal{V}(r)$ denote the set of vertices of r. We also extend this definition to $\mathcal{V}((\ell, r)) = \mathcal{V}(r)$.

With any path π of the region automaton, we associate a labeled bipartite graph $\Gamma(\pi)$ called the *folded orbit graph of* π [18] (FOG for short). Intuitively, the FOG of a path gives the reachability relation between the vertices of the first and the last regions, assuming the guards are closed. For any path π from state q to q', the node set of the graph $\Gamma(\pi)$ is defined as the disjoint union of $\mathcal{V}(q)$ and $\mathcal{V}(q')$. There is an edge from $v \in \mathcal{V}(q)$ to $v' \in \mathcal{V}(q')$, if, and only if v' is reachable from v along the path π when all guards are replaced by their closed counterparts. It was shown that any run along π can be written as a convex combination of runs along vertices; using this observation orbit graphs can be used to characterize runs along given paths [18]. An important property that we will use is that there is a monoid morphism from paths to orbit graphs. In fact, the orbit graph of a path can be obtained by combining the orbit graphs of a factorization of the path.

When the path π is a cycle around q, then $\Gamma(\pi)$ is defined on the node set $\mathcal{V}(q)$, by merging the nodes of the bipartite graph corresponding to the same vertex. A cycle π is *aperiodic* if for all $k \geq 1$, $\Gamma(\pi^k)$ is strongly connected. Aperiodic cycles are closely related to cycles whose FOG is a complete graph since a long enough iteration of the former gives a complete FOG and conversely, any cycle that has some power with a complete FOG is aperiodic (Appendix B Lemma 21). In the timed automaton of Fig. 1(b), the cycles around locations ℓ_2 and ℓ_3 are aperiodic while that of ℓ_1 is not. Appendix A contains examples. Complete FOG are of particular interest to us as they exactly correspond to paths whose reachability relation (between valuations of the initial and last region) is complete [3]. This means that there is no convergence phenomena along the path.

DBMs and shrunk DBMs. We assume the reader is familiar with the data structure of *difference-bound matrix (DBM)* which are square matrices over $(\mathbb{R} \times \{<, \leq\}) \cup \{(\infty, <)\}$ used to represent zones. DBMs were introduced in [6, 13] for analyzing timed automata; see also [5]. Standard operations used to explore the state space of timed automata have been defined on DBMs: intersection is written $M \cap N$, $\mathsf{Pre}\,(M)$ is the set of time predecessors of M, $\mathsf{Unreset}_R(M)$ is the

[5] Note this slight modification in the definition of the region automaton.

set of valuations that end in M when the clocks in R are reset. We also consider $\mathsf{Pre}_{>\delta}(M)$, time predecessors with a delay of more than δ.

To analyze the parametric game $\mathcal{G}_\delta(\mathcal{A})$, we need to express *shrinkings* of zones, *e.g.* sets of states satisfying constraints of the form $g = 1 + \delta < x < 2 - \delta \wedge 2\delta < y$, where δ is a parameter. Formally, a shrunk DBM is a pair (M, P), where M is a DBM, and P is a nonnegative integer matrix called a *shrinking matrix* (SM). This pair represents the set of valuations defined by the DBM $M - \delta P$, for any $\delta > 0$. For instance, M is the guard g obtained by setting $\delta = 0$, and P is made of the integer multipliers of δ.

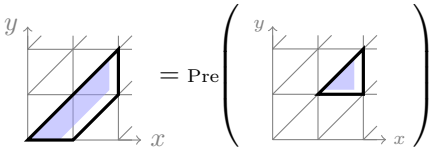

Fig. 2. Time-predecessors operation $(M, P) = \mathsf{Pre}(N, Q)$ applied on a shrunk DBM. Here, the shaded area on the left represents the set $M - \delta P$, while the zone with the thick contour represents M.

We adopt the following notation: when we write a statement involving a shrunk DBM (M, P), we mean that for some $\delta_0 > 0$, the statement holds for $(M - \delta P)$ for all $\delta \in [0, \delta_0]$. For instance, $(M, P) = \mathsf{Pre}_{>\delta}(N, Q)$ means that $M - \delta P = \mathsf{Pre}_{>\delta}(N - \delta Q)$ for all small enough $\delta > 0$. Additional operations are defined for shrunk DBMs: for any (M, P), we define $\mathsf{shrink}_{[-\delta,\delta]}(M, P)$ as the set of valuations ν with $\nu + [-\delta, \delta] \subseteq M - \delta P$, for small enough $\delta > 0$. Figure 2 shows an example of a shrunk DBM and an operation applied on it. Standard operations on zones can also be performed on shrunk DBMs in poly-time [20, 8].

4 Playing in the Region Automaton

In this section, we will define an appropriate abstraction based on region automata in order to characterize winning in the robust timed game. We note that the usual region automaton does not carry enough information for our purpose; for instance, the blocking behavior in Fig.1(a) cannot be detected in the region automaton (which does contain infinite runs). We therefore define, on top of the usual region construction, a complex winning condition $\mathcal{W}$ characterizing accepting runs *along aperiodic cycles*. In order to be able to transfer the condition $\mathcal{W}$ to the continuous semantics, we study the properties of $\mathcal{W}$ on the abstract region game, and derive two necessary and sufficient conditions ($\mathcal{C}_C$ and $\mathcal{C}_P$) for winning which will be used in the next section to derive the algorithm.

Abstract Arena and Strategies We fix a timed automaton $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \ell_0, \mathsf{Act}, E)$ and a Büchi condition ϕ. We define a two-player turn-based game played on the region automaton $\mathcal{R}(\mathcal{A})$. In this game, Controller's strategy consists in choosing actions, while Perturbator's strategy consists in resolving non-determinism.

We consider standard notions of (finite-memory, memoryless) strategies in this game and, given a finite-memory strategy σ, we denote by $\mathcal{R}(\mathcal{A})[\sigma]$ the automaton obtained under strategy σ.

Winning Condition on $\mathcal{R}(\mathcal{A})$ We define set $\mathcal{W}$ of winning plays in the game $\mathcal{R}(\mathcal{A})$: an infinite play is winning iff the following two conditions are satisfied: 1) an accepting state in ϕ is visited infinitely often 2) disjoint finite factors with complete folded orbit graphs are visited infinitely often.

Proposition 2. *The game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ is determined, admits finite-memory strategies for both players, and wining strategies can be computed in EXPTIME.*

The above proposition is proved by showing that condition 2) of $\mathcal{W}$ can be rewritten as a Büchi condition: the set of folded orbit graphs constitute a finite monoid (of exponential size) which can be used to build a Büchi automaton encoding condition 2). Using a product construction for Büchi automata, one can define a Büchi game of exponential size where winning for any player is equivalent to winning in $(\mathcal{R}(\mathcal{A}), \mathcal{W})$. See Appendix B.

From the abstract game to the robust timed game We introduce two conditions for Perturbator and Controller which are used in Section 5 to build concrete strategies in the robust timed game.

$\mathcal{C}_P$: there exists a finite memory strategy τ for Perturbator such that no cycle in $\mathcal{R}(\mathcal{A})[\tau]$ reachable from the initial state is winning aperiodic.
$\mathcal{C}_C$: there exists a finite-memory strategy σ for Controller such that every cycle in $\mathcal{R}(\mathcal{A})[\sigma]$ reachable from the initial state is winning aperiodic.

Intuitively, determinacy allows us to write that either all cycles are aperiodic, or none is, respectively under each player's winning strategies. We prove that these properties are sufficient and necessary for respective players to win $(\mathcal{R}(\mathcal{A}), \mathcal{W})$:

Lemma 3. *The winning condition $\mathcal{W}$ is equivalent to $\mathcal{C}_P$ and $\mathcal{C}_C$: 1.* Perturbator *wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ iff property $\mathcal{C}_P$ holds. 2.* Controller *wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ iff property $\mathcal{C}_C$ holds. In both cases, a winning strategy for $\mathcal{W}$ is also a witness for $\mathcal{C}_C$ (resp. $\mathcal{C}_P$).*

The proof is obtained by the following facts: finite-memory strategies are sufficient to win the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$, thanks to the previous proposition; given a folded orbit graph γ, there exists n such that γ^n is complete iff γ is aperiodic; last, the concatenation of a complete FOG with an arbitrary FOG is complete.

5 Solving the Robust Timed Game

In this section, we show that condition $\mathcal{C}_P$ (resp. $\mathcal{C}_C$) is sufficient to witness the existence of a winning strategy in the robust timed game for Perturbator (resp. Controller). By Lemma 3, the robust timed game problem is then reduced to $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ and we obtain:

Theorem 4. *Let $\mathcal{A}$ be a timed automaton with a Büchi condition. Then,* Controller *wins the robust timed game for $\mathcal{A}$ iff he wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$.*

By Proposition 2, the robust timed game can be solved in EXPTIME. In addition, when Controller wins the robust timed game, one can also compute $\delta > 0$ and an actual strategy in $\mathcal{G}_\delta(\mathcal{A})$: Lemma 3 gives an effective strategy σ satisfying $\mathcal{C}_C$ and the proof of Lemma 6 will effectively derive a strategy (given as shrunk DBMs).

5.1 Sufficient condition for Perturbator

We first prove that under condition $\mathcal{C}_P$, Perturbator wins the robust timed game. We use the following observations. Once one fixes a strategy for Perturbator satisfying $\mathcal{C}_P$, intuitively, one obtains a timed automaton (where there is no more non-determinism in actions), such that all accepting cycles are non-aperiodic. As Perturbator has no more non-determinism to resolve, results of [21] apply and the next lemma follows (see Appendix C).

Lemma 5. *If $\mathcal{C}_P$ holds, then* Perturbator *wins the robust timed game.*

5.2 Sufficient condition for Controller

Proving that $\mathcal{C}_C$ is a sufficient condition for Controller is the main difficulty in the paper; the proof for the next lemma is given in this section.

Lemma 6. *If $\mathcal{C}_C$ holds, then* Controller *wins the robust timed game.*

Proof outline. We consider the non-deterministic automaton $\mathcal{B} = \mathcal{R}(\mathcal{A})[\sigma]$ which represents the behavior of game $\mathcal{R}(\mathcal{A})$ when Controller plays according to σ, given by condition $\mathcal{C}_C$. Without loss of generality, we assume that σ is a *memoryless strategy* played on the game $\mathcal{R}(\mathcal{A})[\sigma]$ (states of $\mathcal{R}(\mathcal{A})$ can be augmented with memory) and that $\mathcal{B}$ is trim. Given an edge $e = q \to q'$ of $\mathcal{B}$, we denote by $\mathsf{edge}(e)$ the underlying transition in $\mathcal{A}$.

Given a state p of $\mathcal{B}$, we denote by $\mathsf{Unfold}(\mathcal{B}, p)$ the infinite labeled tree obtained as the unfolding of $\mathcal{B}$, rooted in state p. Formally, nodes are labeled by states of $\mathcal{B}$ and given a node x labeled by q, $\sigma(q)$ is defined and there exists q' such that $q \xrightarrow{\sigma(q)} q'$ in $\mathcal{B}$. Then x has one child node for each such q'. We may abusively use nodes to refer to labels to simplify notations.

We first choose states $q_1, \ldots, q_n$ such that every cycle of $\mathcal{B}$ contains one of the q_i's. Let us denote by q_0 the initial state of $\mathcal{B}$, for $i = 0..n$, one can choose a finite prefix t_i of $\mathsf{Unfold}(\mathcal{B}, q_i)$ such that every leaf of t_i is labeled by some q_j, $j = 1..n$. Indeed, as $\mathcal{B}$ is trim and σ is a winning strategy for Controller in the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$, every branch of $\mathsf{Unfold}(\mathcal{B}, q_i)$ is infinite.

Strategies for standard timed games can be described by means of regions. In our robustness setting, we use shrinkings of regions. Let (ℓ_i, r_i) be the label of state q_i. To build a strategy for Controller, we will identify $\delta > 0$ and zones s_i, $i = 1..n$, obtained as shrinking of regions r_i. These zones satisfy that the controllable predecessors through the tree t_i computed with zones $(s_j)_j$ at leafs contains the zone s_i: this means that from any configuration in (ℓ_i, s_i), Controller has a strategy to ensure reaching a configuration in one of the (ℓ_j, s_j)'s, when following the tree t_i. These strategies can thus be repeated, yielding infinite outcomes. This idea is illustrated in Fig. 3 where the computations along some prefix t are depicted: the shrunk zone at a node represents the controllable predecessors of the shrunk

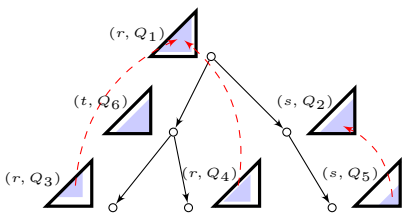

Fig. 3. Proof idea of Lemma 6. Dashed arrows represent cycles.

zones of its children. Here, from the shrunk set of the root node, one can ensure reaching a shrinking of each leaf which is included in the shrinking of the starting state of its cycle, yielding a kind of fixpoint. We have in fact $(r, Q_i) \subseteq (r, Q_1)$ for $i = 3, 4$, and $(s, Q_5) \subseteq (s, Q_2)$.

To identify these zones s_i, we will successively prove the three following facts:
1) Prefixes t_i's can be chosen such that every branch has a complete FOG
2) Controllable predecessors through t_i's of non-empty shrunk zones are non-empty shrunk zones
3) Controllable predecessors through t_i's can be faithfully over-approximated by the intersection of controllable predecessors through branches of t_i

Ensuring branches with complete FOGs. To prove property 1) of the Proof outline, we use condition $\mathcal{C}_C$ and the fact that long enough repetitions of aperiodic cycles yield complete FOGs. We obtain:

Lemma 7. *Under condition $\mathcal{C}_C$, there exists an integer N such that every path ρ in $\mathcal{B}$ of length at least N has a complete folded orbit graph.*

Controllable Predecessors and Merge. In order to compute winning states in $\mathcal{G}_\delta(\mathcal{A})$ through unfoldings, we define two operators. CPre is the standard set of controllable predecessors along a single edge:

Definition 8 (Operator CPre). *Let $e = q \to q_1$ be an edge in some unfolding of $\mathcal{B}$. Let us write $q = (\ell, r)$, $q_1 = (\ell_1, r_1)$, $\sigma(q) = (r', a)$ and $\mathsf{edge}(q \to q_1) = (\ell, g_1, a, R_1, \ell_1)$. Let M_1 be a DBM such that $M_1 \subseteq r_1$ and $\delta \geq 0$. We define the set of δ-controllable predecessors of M_1 through edge e as*
$\mathsf{CPre}_e^\delta(M_1) = r \cap \mathsf{Pre}_{>\delta}\left(\mathsf{Shrink}_{[-\delta,\delta]}\left(r' \cap \mathsf{Unreset}_{R_1}(M_1)\right)\right)$.

The above definition is extended to paths. Intuitively, $\mathsf{CPre}_e^\delta(M_1)$ are the valuations in region r from which M_1 can be surely reached through a delay in r' and the edge e despite perturbations up to δ.

We now consider the case of branching paths, where $\mathsf{Perturbator}$ resolves non-determinism. In this case, in order for $\mathsf{Controller}$ to ensure reaching given subsets in all branches, one needs a stronger operator, which we call CMerge. Intuitively, $\mathsf{CMerge}_{e_1,e_2}^\delta(M_1, M_2)$ is the set of valuations in the region starting r from which $\mathsf{Controller}$ can ensure reaching either M_1 or M_2 by a *single* strategy, whatever $\mathsf{Perturbator}$'s strategy is. The operator is illustrated in Fig. 4.

Definition 9 (Operator CMerge). *Let $e_1 = q \to q_1$ and $e_2 = q \to q_2$ be edges in some unfolding of $\mathcal{B}$, and write $q = (\ell, r)$, $\sigma(q) = (r', a)$ and for $i \in \{1, 2\}$, $q_i = (\ell_i, r_i)$, $\mathsf{edge}(q \to q_i) = (\ell, g_i, a, R_i, \ell_i)$. Let M_i be a DBM such that $M_i \subseteq r_i$ for $i \in \{1, 2\}$. For any $\delta \geq 0$, define the set of δ-controllable predecessors of M_1, M_2 through edges e_1, e_2 as* $\mathsf{CMerge}_{e_1,e_2}^\delta(M_1, M_2) = r \cap \mathsf{Pre}_{>\delta}\left(\mathsf{Shrink}_{[-\delta,\delta]}\left(r' \cap \bigcap_{i\in\{1,2\}} \mathsf{Unreset}_{R_i}(M_i)\right)\right)$.

We extend CMerge by induction to finite prefixes of unfoldings of $\mathcal{B}$ (see Appendix C.2). Informally, consider a tree t and shrunk DBMs $(M_i, P_i)_i$ for its leaves, $\mathsf{CMerge}_t^\delta\big((M_i, P_i)_i\big)$ is the set of states for which there is a strategy ensuring reaching one of (M_i, P_i). It should be clear that because $\mathsf{CMerge}_{e_1,e_2}^\delta$

is more restrictive than $\mathsf{CPre}^\delta_{e_1} \cap \mathsf{CPre}^\delta_{e_2}$, we always have $\mathsf{CMerge}^\delta_t \subseteq \bigcap_\beta \mathsf{CPre}^\delta_\beta$, where β ranges over all branches of t (See Fig. 4).

The following lemma states property 2) of the proof outline. Existence of the SM Q follows from standard results on shrunk DBMs. Non-emptiness of (M, Q) follows from the fact that every delay edge leads to a *non-punctual* region. Define a *full-dimensional subset of a set* $r \subseteq \mathbb{R}^n$ is a subset $r' \subseteq r$ such that there is $\nu \in r'$ and $\varepsilon > 0$ satisfying $\mathsf{Ball}_{d_\infty}(\nu, \varepsilon) \cap r \subseteq r'$, where $\mathsf{Ball}_{d_\infty}(\nu, \varepsilon)$ is the standard ball of radius ε for the infinity norm on $\mathbb{R}^n$.

Lemma 10. *Let t be a finite prefix of $\mathsf{Unfold}(\mathcal{B}, q)$, r the region labeling the root, and $r_1, \ldots, r_k$ those of the leafs. $M, N_1, \ldots, N_k$ be non-empty DBMs that are full dimensional subsets of $r, r_1, \ldots, r_k$ satisfying $M = \mathsf{CMerge}^0_t((N_j)_j)$. We consider shrinking matrices P_j, $1 \leq j \leq k$, of DBM N_j such that $(N_j, P_j) \neq \emptyset$. Then, there exists a SM Q such that $(M, Q) = \mathsf{CMerge}^\delta_t((N_j, P_j)_j)$, and $(M, Q) \neq \emptyset$.*

Over-Approximation of CMerge. Given a prefix t where each branch β_i ends in a leaf labeled with (r_i, P_i), we see $\cap_{\beta_i} \mathsf{CPre}^0_{\beta_i}((r_i, P_i))$ as an over-approximation of $\mathsf{CMerge}^0_t((r_i, P_i)_i)$. We will show that both sets have the same "shape", *i.e.* any facet that is not shrunk in one set, is not shrunk in the other one. This is illustrated in Fig. 4.

We introduce the notion of *0-dominance* as follows: for a pair of SMs P, Q, Q *0-dominates* P, written $P \preceq_0 Q$, if $Q[i,j] \leq P[i,j]$, and $Q[i,j] = 0$ implies $P[i,j] = 0$ for all i, j. Informally, a set shrunk by P is smaller than that shrunk by Q, but yields the same shape. The 0-dominance is the notion we use for a "precise" over-approximation of CMerge: (see Appendix C.3)

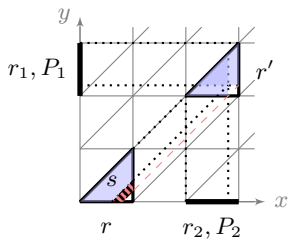

Fig. 4. We have $s = \mathsf{CMerge}^0((r_i, P_i)_i)$, which is strictly included in $\cap_i \mathsf{CPre}^0(r_i, P_i)$ but has the same shape.

Lemma 11. *Let t be a finite prefix of $\mathsf{Unfold}(\mathcal{B}, q)$, with $q = (\ell, r)$, and let (ℓ_i, r_i), $1 \leq i \leq k$ denote the (labels of) leafs of t. We denote by β_i, $1 \leq i \leq k$, the branches of t. Consider SMs P_i, $1 \leq i \leq k$, for regions r_i. Let us denote $(r, P) = \mathsf{CMerge}^0_t((r_i, P_i)_{1 \leq i \leq k})$ and $(r, Q) = \bigcap_{i=1}^k \mathsf{CPre}^0_{\beta_i}(r_i, P_i)$, then $P \preceq_0 Q$.*

Putting everything together. In order to complete the proof of Lemma 6, we first recall the following simple lemma:

Lemma 12 ([21]). *For any DBM M, there is a SM P_0 s.t. $(M, P_0) \neq \emptyset$, and is fully dimensional, and for any SM P and $\varepsilon > 0$ with $(M, P) \neq \emptyset$ and $M - \varepsilon P_0 \neq \emptyset$, we have $M - \varepsilon P_0 \subseteq (M, P)$.*

Remember we have identified states q_i and trees t_i, $i = 0..n$. Denote (ℓ_i, r_i) the label of q_i. For each $i = 1..n$, we denote by P_i the SM obtained by Lemma 12 for r_i. Consider now some tree t_i, $i = 0..n$, with $((r_j, P_j)_j)$ at leafs. Let β_j be a branch of t_i and denote by (r_j, P_j) its leaf. By Lemma 7, the FOG of β_j is complete, and thus from *any* valuation in r_i, one can reach every valuation in the target region r_j along β_j (see [3]), and thus $r_i = \mathsf{CPre}^0_{\beta_j}(r_j, P_j)$. This holds for every branch and we obtain $r_i = \bigcap_j \mathsf{CPre}^0_{\beta_j}(r_j, P_j)$. By Lemma 11, this entails $r_i = \mathsf{CMerge}^0_{t_i}((r_j, P_j)_j)$. We can choose $\varepsilon > 0$ small enough such

that the zone $s_i = r_i - \varepsilon P_i$ is non-empty for every $i = 1..n$ and we obtain $r_i = \mathsf{CMerge}^0_{t_i}((s_j)_j)$. We can then apply Lemma 10, yielding some SM Q_i of r_i such that $\emptyset \neq (r_i, Q_i) = \mathsf{CMerge}^\delta_{t_i}((s_j)_j)$. There are two cases:

- $i = 0$: as r_0 is the singleton $\{\mathbf{0}\}$, we have $(r_0, Q_0) = r_0$, and thus $r_0 = \mathsf{CMerge}^\delta_{t_0}((s_j)_j)$. In other terms, for small enough δ's, Controller has a strategy in $\mathcal{G}_\delta(\mathcal{A})$ along t_0 to reach one of the (ℓ_j, s_j)'s starting from the initial configuration $(\ell_0, \mathbf{0})$.
- $i \geq 1$: Lemma 12 entails $s_i \subseteq \mathsf{CMerge}^\delta_{t_i}((s_j)_j)$, which precisely states that for small enough δ's, Controller has a strategy in $\mathcal{G}_\delta(\mathcal{A})$ along t_i, starting in (ℓ_i, s_i), to reach one of the (ℓ_j, s_j)'s.

These strategies can thus be combined and repeated, yielding the result.

6 Probabilistic Semantics

In some systems, considering the environment as a completely adversarial opponent is too strong an assumption. In order to model the environment with more realistic behavior, we define two semantics as probabilistic variants of the robust timed games. The first variant is the *stochastic game semantics* where Perturbator only resolves the non-determinism in actions, but the perturbations are chosen independently and uniformly at random in the interval $[-\delta, \delta]$. The second semantics is the *Markov decision process semantics* (MDP for short), where the non-determinism is also resolved by a uniform distribution on the edges. Note that there is no player Perturbator in the latter semantics.

6.1 Stochastic Game Semantics

Formally, given $\delta > 0$, the state space is partitioned into $V_C \cup V_P$ as previously. At each step, Controller picks a delay $d \geq \delta$, and an action a such that for every edge $e = (\ell, g, a, R, \ell')$ such that $\nu + d \models g$, we have $\nu + d + \varepsilon \models g$ for all $\varepsilon \in [-\delta, \delta]$, and there exists at least one such edge e. Perturbator then chooses an edge e with label a, and a perturbation $\varepsilon \in [-\delta, \delta]$ is chosen independently and uniformly at random. The next state is determined by delaying $d + \varepsilon$ and taking the edge e. To ensure that probability measures exist, we restrict to *measurable strategies*.

In this semantics, we are interested in deciding whether Controller can ensure a given Büchi objective almost surely, for some $\delta > 0$. It turns out that the same characterization as in Theorem 4 holds in the probabilistic case.

Theorem 13. *It is EXPTIME-complete to decide whether for some $\delta > 0$,* Controller *has a strategy achieving a given Büchi objective almost surely in the stochastic game semantics. Moreover, if $\mathcal{C}_C$ holds then* Controller *wins almost-surely; if $\mathcal{C}_P$ holds then* Perturbator *wins almost-surely.*

This is a powerful result showing a strong distinction between robust and non-robust timed games: in the first case, a controller that ensures the specification almost surely can be computed, while in non-robust timed games, *any* controller will fail *almost surely*. Thus, while in previous works on robustness in timed automata (e.g. [18]) the emphasis was on additional behaviors that might appear

in the worst-case due to the accumulation of perturbations, we show that in our setting, this is inevitable. Note that this also shows that limit-sure winning (see next section) is equivalent to almost-sure winning.

6.2 Markov decision process semantics

The *Markov decision process semantics* consists in choosing both the perturbations, and the edges uniformly at random (and independently). Formally, it consists in restricting Perturbator to choose all possible edges uniformly at random in the stochastic game semantics. We denote by $\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A})$ the resulting game, and $\mathbb{P}^\sigma_{\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A}),s}$ the probability measure on $\mathsf{Runs}(\mathcal{A}, s)$ under strategy σ.

For a given timed Büchi automaton, denote ϕ the set of accepting runs. We are interested in the two following problems: (we let $s_0 = (\ell_0, \mathbf{0})$)

Almost-sure winning: does there exist $\delta > 0$ and a strategy σ for Controller such that $\mathbb{P}^\sigma_{\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A}),s_0}(\phi) = 1$?

Limit-sure winning: does there exist, for every $0 \leq \varepsilon \leq 1$, a perturbation upper bound δ, and a strategy σ for Controller such that $\mathbb{P}^\sigma_{\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A}),s_0}(\phi) \geq 1 - \varepsilon$?

Observe that if almost-sure winning cannot be ensured, then limit-sure winning still has a concrete interpretation in terms of controller synthesis: given a quantitative constraint on the quality of the controller, what should be the precision on clocks measurements to be able to synthesize a correct controller? Consider the timed automaton depicted on the right. It is easy to see that Controller loses the (non-stochastic) robust game, the stochastic game and in the MDP semantics with almost-sure condition, but he wins in the MDP semantics with limit-sure condition.

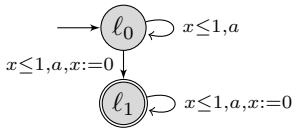

Fig. 5. This automaton is losing in the MDP semantics for the almost-sure winning but winning under the same semantics for the limit-sure winning. In fact, a blocking state (ℓ_0, x) with $x > 1 - \delta$ is reachable with positive probability for any δ.

Theorem 14. *It is EXPTIME-complete to decide whether* Controller *wins almost-surely (resp. limit-surely) in the MDP semantics of a timed Büchi automaton.*

To prove this theorem, we will define decidable characterizations on $\mathcal{R}(\mathcal{A})$ which we will see as a finite Markov decision process. In this MDP, the nondeterminism of actions is resolved according to a uniform distribution. Given a strategy $\hat{\sigma}$ for Controller and a state v, we denote by $\mathbb{P}^{\hat{\sigma}}_{\mathcal{R}(\mathcal{A}),v}$ the resulting measure on $\mathsf{Runs}(\mathcal{R}(\mathcal{A}), v)$. The initial state of $\mathcal{R}(\mathcal{A})$ is v_0. We will use well-known notions about finite MDPs; we refer to [19].

Almost-sure winning We introduce the following winning condition $\mathcal{W}'$: a Controller's strategy $\hat{\sigma}$ in $\mathcal{R}(\mathcal{A})$ is winning in state v iff $\mathbb{P}^{\hat{\sigma}}_{\mathcal{R}(\mathcal{A}),v}(\phi) = 1$ and every run in $\mathsf{Runs}(\mathcal{R}(\mathcal{A})[\hat{\sigma}], v)$ contains infinitely many disjoint factors whose FOG is complete. Observe that this combines an almost-sure requirement with a sure requirement. This winning condition is our characterization for almost-sure winning:

Proposition 15. Controller *wins almost-surely in the MDP semantics of a timed Büchi automaton* $\mathcal{A}$ *iff* Controller *wins the game* $(\mathcal{R}(\mathcal{A}), \mathcal{W}')$ *in* v_0.

Intuitively, the first condition is required to ensure winning almost-surely, and the second condition allows to forbid blocking behaviors. Notice the resemblance with condition $\mathcal{W}$; the difference is that ϕ only needs to be ensured almost-surely rather than surely. We prove the decidability of this condition.

Lemma 16. *The game $(\mathcal{R}(\mathcal{A}), \mathcal{W}')$ admits finite-memory strategies, and winning strategies can be computed in EXPTIME.*

The proof of Prop. 15 uses the following ideas. We first assume that Controller wins the abstract game using some strategy $\hat{\sigma}$. We derive from $\hat{\sigma}$ a strategy σ in the MDP semantics by concretizing the delays chosen by σ. To do so, we consider the automaton $\mathcal{R}(\mathcal{A})[\hat{\sigma}]$ and proceed as in Section 5, which results in a strategy defined by means of shrinking matrices. Using the results of Section 5, we can prove that the outcomes of σ are never blocked, and thus the probabilities of paths in $\mathcal{R}(\mathcal{A})$ under $\hat{\sigma}$ are preserved by σ. As a consequence, σ wins almost-surely.

Conversely, by contradiction, we assume that Controller does not satisfy $\mathcal{W}'$ in $\mathcal{R}(\mathcal{A})$ while there exists an almost-sure strategy σ for the MDP semantics. We build from σ a strategy $\hat{\sigma}$ in $\mathcal{R}(\mathcal{A})$, and prove that it satisfies ϕ almost-surely. This entails the existence of a run ρ in $\mathcal{R}(\mathcal{A})[\hat{\sigma}]$ such that ρ eventually does not contain factors with a complete FOG. We finally show that, with positive probability, perturbations can be chosen to ensure that the run gets blocked along a finite prefix of this path, Lemma 43), which ensures that σ is not almost-surely winning.

Limit-sure winning As illustrated in Fig. 5, it is sometimes possible, according to any $\varepsilon > 0$, to choose the parameter $\delta > 0$ small enough to ensure a winning probability of at least $1 - \varepsilon$. The idea is that in such cases one can ensure reaching the set of almost-sure winning states with arbitrarily high probability, although the run can still be blocked with a small probability before reaching this set.

To characterize limit-sure winning, we define a new condition $\mathcal{W}''$ as follows. If $\textsc{Win}'$ denotes the set of winning states for Controller in the game $(\mathcal{R}(\mathcal{A}), \mathcal{W}')$, then $\mathcal{W}''$ is defined as the set of states from which one can almost surely reach $\textsc{Win}'$.

Proposition 17. Controller *wins limit-surely in the MDP semantics of a timed Büchi automaton $\mathcal{A}$ from s_0 iff* Controller *wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W}'')$ in v_0.*

The proof of this proposition relies on the following lemma, and uses techniques similar as those introduced to prove Proposition 15.

Lemma 18. *The game $(\mathcal{R}(\mathcal{A}), \mathcal{W}'')$ admits finite-memory strategies, and winning strategies can be computed in EXPTIME.*

7 Conclusion

In this paper, we defined robust timed games with Büchi conditions and unknown imprecision parameters. Our formalism allows one to solve robust controller synthesis problems both against an adversarial (or worst-case) environment, and two variants of probabilistic environments. The procedures we have developed

allow, when they exist, to effectively build a bound $\delta > 0$ on the perturbation and a winning strategy for Controller. Some questions remain open including the generalization of these results to concurrent timed games with parity conditions considered in [11]. We believe it is possible to derive symbolic algorithms but this will require extending the theory to detect aperiodic cycles in zone graphs rather than in the region graph.

References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. Eugene Asarin, Oded Maler, Amir Pnueli, and Joseph Sifakis. Controller synthesis for timed automata. In SSC'98, p. 469–474. Elsevier Science, 1998.
3. Nicolas Basset and Eugene Asarin. Thin and thick timed regular languages. In FORMATS'11, LNCS 6919, p. 113–128. Springer, 2011.
4. Danièle Beauquier. On probabilistic timed automata. *Theor. Comput. Sci.*, 292(1):65–84, January 2003.
5. Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, LNCS 2098, p. 87–124. Springer, 2004.
6. Bernard Berthomieu and Miguel Menasche. An enumerative approach for analyzing time Petri nets. In WCC'83, p. 41–46. North-Holland/IFIP, September 1983.
7. Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. Robust analysis of timed automata via channel machines. In FoSSaCS'08, LNCS 4962, p. 157–171. Springer, 2008.
8. Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robust reachability in timed automata: A game-based approach. In ICALP'12, LNCS 7392, p. 128–140. Springer, 2012.
9. Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robustness in timed automata. In RP'13, LNCS 8169, p. 1–18. Springer, 2013.
10. Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR 2005*, p. 66–80. Springer, 2005.
11. Krishnendu Chatterjee, Thomas A. Henzinger, and Vinayak S. Prabhu. Timed parity games: Complexity and robustness. *Logical Methods in Computer Science*, 7(4), 2011.
12. Martin De Wulf, Laurent Doyen, Nicolas Markey, and Jean-François Raskin. Robust safety of timed automata. *Formal Methods in System Design*, 33(1-3):45–84, 2008.
13. David L. Dill. Timing assumptions and verification of finite-state concurrent systems. In AVMFSS'89, LNCS 407, p. 197–212. Springer, 1990.
14. Vojtęch Forejt, Marta Kwiatkowska, Gethin Norman, and Ashutosh Trivedi. Expected reachability-time games. In *Formal Modeling and Analysis of Timed Systems*, LNCS 6246, p. 122–136. Springer Berlin Heidelberg, 2010.
15. Thomas A. Henzinger and Joseph Sifakis. The embedded systems design challenge. In FM'06, LNCS 4085, p. 1–15, Hamilton, Canada, 2006. Springer.
16. Henrik E Jensen. Model checking probabilistic real time systems. In *Proc. 7th Nordic Workshop on Programming Theory*, p. 247–261. Citeseer, 1996.
17. Marta Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theor. Comput. Sci.*, 282(1):101–150, June 2002.
18. Anuj Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10(1-2):87–113, 2000.

19. Martin L. Putterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, New York, NY, 1994.
20. Ocan Sankur, Patricia Bouyer, and Nicolas Markey. Shrinking timed automata. In FSTTCS'11, LIPIcs 13, p. 375–386. Leibniz-Zentrum für Informatik, 2011.
21. Ocan Sankur, Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. Robust controller synthesis in timed automata. In CONCUR'13, LNCS 8052, p. 546–560. Springer, 2013.

A Orbit Graphs

We start with some complements on the region automaton $\mathcal{R}(\mathcal{A})$. We may sometimes decompose edges $(\ell, r) \xrightarrow{(r',a)} (\ell', s)$, as the two following edges: $(\ell, r) \xrightarrow{\Delta} (\ell, r')$ denotes the delay edge, which represents time-elapsing in the timed automaton, and $(\ell, r') \xrightarrow{e} (\ell', s)$ with $e = (\ell, g, a, R, \ell') \in E$ is the edge of $\mathcal{A}$ used to build the transition $(\ell, r) \xrightarrow{(r',a)} (\ell', s)$ of $\mathcal{R}(\mathcal{A})$. This representation of edges in $\mathcal{R}(\mathcal{A})$ is thus more precise. We will use it in the appendices.

Using this notation, given a run $\rho = p_1 e_1 p_2 e_2 \dots p_n$ of $\mathcal{A}$ with $e_i \in \mathbb{R}_{>0} \cup E$, its *projection on regions* is the path $\pi = q_1 e'_1 q_2 e'_2 \dots q_n$ in the region automaton s.t. $p_i \in q_i$, and $e'_i = e_i$ if $e_i \in E$ and $e'_i = \Delta$ otherwise.

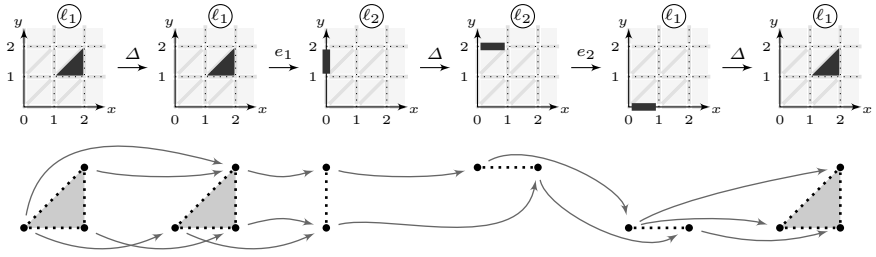

Fig. 6. The orbit graph of a (cyclic) path in the region automaton of the automaton of Fig. 1(a).

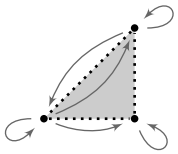

Fig. 7. The folded orbit graph of the (non-forgetful) cycle of Fig. 6.

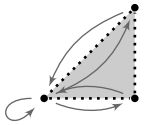

Fig. 8. The folded orbit graph of a forgetful cycle.

With any path π of the region automaton, we associate a labeled bipartite graph $\Gamma(\pi)$ called the *folded orbit graph of* π [8] (FOG for short). Intuitively, the FOG of a path gives the reachability relation between the vertices of the first and last regions. Formally, for a transition $\tau = q_1 e_1 q_2$, its orbit graph $\Gamma(\tau) = (V_1 \cup V_2, (q_1, q_2), E)$ is a bipartite graph where $V_1 = \{(1, v)\}_{v \in \mathcal{V}(q_1)}$, and $V_2 = \{(2, v)\}_{v \in \mathcal{V}(q_2)}$. For any $\big((1, u), (2, v)\big) \in V_1 \times V_2$, we have an edge $((1, u), (2, v)) \in E$, if, and only if $u \xrightarrow{\overline{e_1}} v$, where $\overline{e_1} = \Delta$ if $e_1 = \Delta$, and otherwise $\overline{e_1}$ is obtained by replacing the guard by its closed counterpart. Note that each vertex has at least one successor through $\overline{e_1}$ [1]. In order to extend $\Gamma(\cdot)$ to paths, we use a composition operator $\oplus$ between FOGs, defined as follows. If $G = (V_1 \cup V_2, (q_1, q_2), E)$ and $G' = (V'_1 \cup V'_2, (q'_1, q'_2), E')$ denote two FOGs, then $G \oplus G'$ is defined if, and only if, $q_2 = q'_1$, that is, when the path defining the former graph ends in the first state of the path defining the latter graph. In this case, the graph $G'' = G \oplus G' = (V_1 \cup \dots V'_2, (q_1, q'_2), E'')$ is defined by taking

the disjoint union of G and G', merging each node $(2, v)$ of V_2 with the node $(1, v)$ of V_1'. Now, we extend $\Gamma(\cdot)$ to paths by induction, as follows. Consider any path $\pi = q_1 e_1 \dots q_{n-1} e_{n-1} q_n$, and let $G = (V_1 \cup V_2, (q_1, q_{n-1}), E)$ be the FOG $\Gamma(q_1 e_1 \dots q_{n-1})$, given by induction. Let $G' = (U \cup U', (q_{n-1}, q_n), E')$ denote the bipartite graph of $q_{n-1} e_{n-1} q_n$. Then, we let $\Gamma(\pi) = G \oplus G'$. If the given path $\pi = q_1 \dots q_1$ is a cycle, then we define $\Gamma(\pi)$ on the node set $\mathcal{V}(q_1)$, by merging the nodes of the bipartite graph corresponding to the same vertex. Note that delays of duration zero are allowed when defining orbit graphs. Figure 6 displays a path in the region automaton of the automaton depicted on Fig. 1(a). Figure 7 shows the FOG of this path.

Orbit graphs are used to characterize runs along a given path on regions:

Lemma 19 ([8]). *Let π be a path from region r to s. Consider any $\nu \in r$ with $\nu = \sum_{v \in \mathcal{V}(r)} \lambda_v v$ for some coefficients $\lambda_v \geq 0$ and $\sum \lambda_v = 1$. If $\nu \xrightarrow{\pi} \nu'$, then for each $v \in \mathcal{V}(r)$, there exists a probability distribution $\{p_{v,w}^{\nu,\nu'}\}_{w \in R_{\Gamma(\pi)}(v)}$ over $R_{\Gamma(\pi)}(v)$ such that $\nu' = \sum_{v \in \mathcal{V}(r)} \lambda_v \sum_{w \in R_{\Gamma(\pi)}(v)} p_{v,w}^{\nu,\nu'} w$.*[6]

B Proofs of Section 4 (Playing in the Region Automaton)

In order to prepare the proof of Prop. 2, we first construct a deterministic Büchi automaton $\mathcal{B}$ recognizing runs of $\mathcal{R}(\mathcal{A})$ that contain an infinite number of disjoint factors with complete FOGs.

The goal of $\mathcal{B}$ is to take track of the current folded orbit graph, and recognize complete orbit graphs. The state space contains all G and triples (G, a, r) where G ranges over all folded orbit graphs of $\mathcal{A}$, r is a region, and a a label. The size of the state space is thus bounded by an exponential (see Section 3); the initial state is $\Gamma(\emptyset)$ (that is, the graph of the identity relation). Informally, the transitions implement the operation $\oplus$ on the monoid of folded orbit graphs. More precisely, we have $G \xrightarrow{a,r'} (G, a, r')$, and $(G, a, r') \xrightarrow{e} (G \oplus \Gamma((\ell, r)\Delta(\ell, r')e(\ell', s)))$, where $(\ell, r)\Delta(\ell, r')e(\ell', s)$ is the path generated by delaying to r' and taking the edge e. More precisely, if we write $e = (\ell, g, a, R, \ell')$, then $s = r'[R \leftarrow 0]$. All states G such that G is a complete FOG are accepting. Furthermore, from any such accepting state G, we remove the previously defined outgoing edges, and add the edges $G \xrightarrow{a,r'} (\Gamma(\emptyset), a, r')$. Hence, the automaton tracks the FOG of the current factor it is reading, and reinitializes the FOG when it visits a complete graph. Note that $\mathcal{B}$ is a deterministic Büchi automaton.

Lemma 20. *Given any timed automaton $\mathcal{A}$, the Büchi automaton $\mathcal{B}$ constructed as above is deterministic and recognizes the set of runs of $\mathcal{R}(\mathcal{A})$ that contain an infinite number of disjoint factors with complete FOGs.*

Proposition 2. *The game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ is determined, admits finite-memory strategies for both players, and wining strategies can be computed in EXPTIME.*

Proof. We first define a game $\mathcal{B}_1$ as a slight modification of $\mathcal{R}(\mathcal{A})$ where the moves of both players are made explicit, in order to allow parallel composition

[6] $R_{\Gamma(\pi)}(v)$ denotes here the successors of v in $\Gamma(\pi)$.

with the automaton $\mathcal{B}$ we just defined. Given a timed automaton $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \ell_0, \Sigma, E)$ and a Büchi condition B, the state space of $\mathcal{B}_1$ is defined as Player 1 states (ℓ, r) and Player 2 states (ℓ, r, r', a), where ℓ is a location, r and r' regions, and a a label of $\mathcal{A}$. The labels of $\mathcal{B}_1$ are either pairs (r, a) where r is a region, and a is the label of an action, or edges $e \in E$. The transitions are defined as follows. We have $(\ell, r) \xrightarrow{(r',a)} (\ell, r, r', a)$ iff $(\ell, r) \xrightarrow{(r',a)} (\ell', s)$ in $\mathcal{R}(\mathcal{A})$ for some state (ℓ', s). For any edge $e = (\ell, g, a, R, \ell')$ with label a, such that g is satisfied by r', we let $(\ell, r, r', a) \xrightarrow{e} (\ell', s)$ where $s = r'[R \leftarrow 0]$. Hence, these transitions "simulate" those of $\mathcal{R}(\mathcal{A})$ while the moves of both players are made visible at intermediate states. As before, the accepting states are (ℓ, r) such that $\ell \in B$. The initial state is $\{(\ell_0, r_0)\}$.

Now, let $\mathcal{B} \parallel \mathcal{B}_1$ denote the parallel composition of $\mathcal{B}_1$ and $\mathcal{B}$ endowed with the generalized Büchi condition given by the conditions of the two automata. We already noted that $\mathcal{B}_1$ simply visits the region automaton given moves by both players, and that given the same path, $\mathcal{B}$ keeps track of the orbit graph of the factors. We see $\mathcal{B} \parallel \mathcal{B}_1$ as a game where any state with a Player 1 component in $\mathcal{B}_1$ is a Player 1 state. It is now clear that any winning strategy for $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ can be translated into a winning strategy in $\mathcal{B} \parallel \mathcal{B}_1$ and conversely. We know that in $\mathcal{B} \parallel \mathcal{B}_1$, 1-bit memory strategies are sufficient to win. Given such a strategy, one can derive a strategy for $\mathcal{B}_1$ by seeing $\mathcal{B}$ as the memory (thanks to the fact that it is deterministic). This provides a winning strategy for $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ with at most exponential memory.

Note that the equivalence with a Büchi game means that the game is determined. □

In order to prepare the proof of Lemma 3, we need a couple of results proven in previous works.

The following lemma explains the relation between aperiodic cycles and cycles with complete orbit graphs.

Lemma 21. *If π is an aperiodic cycle, then $\Gamma(\pi^n)$ is a complete graph for all $n \geq |\mathcal{C}_0| \times |\mathcal{C}_0|!$. Conversely, if π is a cycle such that $\Gamma(\pi^n)$ is a complete graph for some $n \geq 1$, then π is aperiodic.*

Proof. For the direct implication, Lemma 13 in [11] actually states that $\Gamma(\pi^n)$ is complete for *some* $n \leq |\mathcal{C}_0| \times |\mathcal{C}_0|!$. To conclude, it suffices to observe that the (left or right) concatenation of a complete folded orbit graph with any folded orbit graph is still complete. Conversely, first note that if $\Gamma(\pi^n)$ is complete for some n, then it is also complete for any $n' \geq n$. Suppose that π is not aperiodic, then there exists k such that the graph $\Gamma(\pi^k)$ is not strongly connected. But then $\Gamma(\pi^{ki})$ is also not strongly connected, for any $i \geq 1$. Contradiction. □

The following lemma is a factorization lemma *à la* Ramsey. Although Ramsey's theorem can be used to prove it, better bounds were obtained in [11].

Lemma 22 ([11]). *Let π be a path of $\mathcal{R}(\mathcal{A})$ written as $\pi = \pi_0\pi_1\pi_2 \dots \pi_n$ where each π_i is a cycle that starts in the same state, for $i \geq 1$. Then, one can write $\pi = \pi'_0\pi'_1\pi'_2 \dots \pi'_{m+1}$ such that $m \geq \sqrt{n/r - 2} - 1$, where $r = 2^{(|\mathcal{C}|+1)^2}|\mathcal{R}(\mathcal{A})|$, and for some indices $0 = \alpha_0 < \alpha_1 < \dots$, we have $\pi'_i = \pi_{\alpha_i} \cdot \dots \cdot \pi_{\alpha_{i+1}-1}$ for each $i \geq 0$, and $\Gamma(\pi'_1) = \Gamma(\pi'_i)$ for all $1 \leq i \leq m$.*

Once a strategy σ for Controller is fixed and a strategy τ for Perturbator is fixed, this defines a unique play ρ in $\mathcal{R}(\mathcal{A})$. We say that ρ is consistent with the couple (σ, τ).

Lemma 3. *The winning condition $\mathcal{W}$ is equivalent to $\mathcal{C}_P$ and $\mathcal{C}_C$: 1.* Perturbator *wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ iff property $\mathcal{C}_P$ holds. 2.* Controller *wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ iff property $\mathcal{C}_C$ holds. In both cases, a winning strategy for $\mathcal{W}$ is also a witness for $\mathcal{C}_C$ (resp. $\mathcal{C}_P$).*

Proof (Proof of Lemma 3).

1. We start by showing the direct implication. Assuming Perturbator wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$, we know by Proposition 2, that there exists a winning finite-memory strategy τ for Perturbator. Let us show that condition $\mathcal{C}_P$ holds. Assume towards a contradiction that there exists a reachable winning aperiodic cycle in $\mathcal{R}(\mathcal{A})[\tau]$. Denote by π such a cycle. This means that Controller has a strategy σ to eventually reach π and repeat π forever. Since π is aperiodic, for some n, we know that $\Gamma(\pi^n)$ is a complete graph (Lemma 21). Thus, the play consistent with (σ, τ) that starts in (ℓ_0, r_0) belongs to $\mathcal{W}$, a contradiction.
   We show the converse implication. Assume that $\mathcal{C}_P$ holds and let us prove that Perturbator wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$. Let τ be the finite memory strategy given by $\mathcal{C}_P$. By contradiction, suppose that there exists a strategy σ for Controller such that the play ρ consistent with the couple of strategies (σ, τ) starting in (ℓ_0, r_0) belongs to $\mathcal{W}$. ρ is a run in the automaton $\mathcal{R}(\mathcal{A})[\tau]$, and therefore one can factorize ρ into $\rho_0\rho_1\cdots$ such that ρ_i for any $i > 0$ is a cycle in $\mathcal{R}(\mathcal{A})[\tau]$ around a same state. Now, as ρ is accepting and there are finitely many folded orbit graphs, one can use a Lemma 22 to factor $\rho_1\rho_2\cdots$ into $\rho'_1\rho'_2\cdots$ such that ρ'_i is an accepting cycle around a same state for $i > 0$, and $\Gamma(\rho'_1) = \Gamma(\rho'_2) = \cdots = \gamma$. Since $\mathcal{C}_P$ holds, γ is not aperiodic. As $\rho \in \mathcal{W}$, there exist $i < j$ such that $\rho'_i \cdots \rho'_j$ contains a factor with a complete FOG. Then we have that $\Gamma(\rho'_i \cdots \rho'_j) = \gamma^{j-i+1}$ is complete, contradiction, as γ is not aperiodic (Lemma 21).
2. Direct implication. Assume that Controller wins in $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ and let us show that $\mathcal{C}_C$ holds. By Proposition 2, there exists a winning finite-memory strategy σ for Controller. Let us show that every reachable cycle in $\mathcal{R}(\mathcal{A})[\sigma]$ is winning aperiodic. Assume towards a contradiction the existence of a cycle which is not winning aperiodic. Denote this cycle by π. This means that there exists a strategy τ for Perturbator such that the cycle π is eventually reached and repeated forever. Because π is not winning aperiodic, the play that starts in (ℓ_0, r_0) and stays consistent with the couple of strategies (σ, τ) cannot belong to $\mathcal{W}$, a contradiction.
   Let us show the converse implication. Assume that $\mathcal{C}_C$ holds, and let σ the finite memory strategy given by $\mathcal{C}_C$. We show that the strategy σ is wining for Controller in $(\mathcal{R}(\mathcal{A}), \mathcal{W})$. Let τ be an arbitrary strategy for Perturbator and let ρ be the play consistent with (σ, ρ). As in case 1., one can factorize ρ into $\rho'_0\rho'_1\rho'_2\cdots$ such that ρ'_i for $i > 0$ are cycles around a same state and $\Gamma(\rho'_1) = \Gamma(\rho'_2) = \cdots = \gamma$. To see that $\rho \in \mathcal{W}$, it suffices to notice that ρ'_i for $i > 1$ is an accepting cycle and that γ is aperiodic, thus there exists n such that γ^n is complete.

□

C Proofs of Section 5 (Solving the Robust Timed Game)

We note the following lemma from [11].

Lemma 23 ([11]). *For a timed automaton $\mathcal{A}$ with distinct labels* Controller *wins the robust timed game in $\mathcal{A}$ iff $\mathcal{R}(\mathcal{A})$ has a reachable winning aperiodic cycle.*

Lemma 5. *If $\mathcal{C}_P$ holds, then* Perturbator *wins the robust timed game.*

Proof. Let τ be Perturbator's finite-memory strategy yielding $\mathcal{R}(\mathcal{A})[\tau]$ witnessing property $\mathcal{C}_P$.

Let $\mathcal{A}'$ be the timed automaton defined on the set of locations $\mathcal{R}(\mathcal{A})[\tau]$, where the clock set is that of $\mathcal{A}$. We have an edge $(\ell, r) \to (\ell, r')$ with the guard r', whenever r' is a time-successor region of r (that is, $\exists d \geq 0, r + d \in r'$). Furthermore, there exists an edge $(\ell, r) \xrightarrow{a} (\ell', s)$ with guard r' if there exists an edge (ℓ, g, a, R, ℓ') with $r' \models g$ in $\mathcal{A}$. By Lemma 23, Perturbator has a winning strategy τ' in $\mathcal{A}'$, since $\mathcal{R}(\mathcal{A}') = \mathcal{R}(\mathcal{A})$, and it has no reachable aperiodic cycles. Now, we show that the combination of τ and τ' is a winning strategy for Perturbator in the robust timed game on $\mathcal{A}$.

We define strategy τ'' in $\mathcal{A}$ as follows. At any state (ℓ, ν), if Controller chooses the delay $d \geq 0$ and action a, then τ'' determines the edge (ℓ, g, a, R, ℓ') following strategy τ, and the perturbation is chosen following τ'. The fact that we respect the strategy τ ensures that the projection of the play to regions stay within $\mathcal{R}(\mathcal{A})[\tau]$. So the projection to regions of any outcome is also a path in $\mathcal{R}(\mathcal{A}')$. The fact that the perturbations are chosen by τ' means that the generated play of $\mathcal{A}$ is also a play of $\mathcal{A}'$. If some play of $\mathcal{A}$ under τ'' were infinite and accepting, this would mean that there is a corresponding play in $\mathcal{A}'$ winning for Controller, which is a contradiction. □

Lemma 7. *Under condition $\mathcal{C}_C$, there exists an integer N such that every path ρ in $\mathcal{B}$ of length at least N has a complete folded orbit graph.*

Proof. The arguments used in this proof already appear in the proof of Lemma 3; we sketch the ideas. It suffices to choose N large enough and apply Lemma 22 (with $n \leftarrow N$). In fact, if all branches of the tree are long enough, then for each branch π, we get a factorization of the form $\pi = \pi'_0\pi'_1\pi'_2 \dots \pi'_{m+1}$ with $m \geq |\mathcal{C}_0| \times |\mathcal{C}_0|!$. By Lemma 21, this means that π contains a factor with a complete FOG. □

C.1 Shrunk DBMs

In order to make the presentation of the paper self-contained, we summarize here some results about shrunk DBMs and give details about elementary operations on this data structure. Most of the content of this section is from [5].

We call elementary operator on shrunk DBMs any operator obtained by mean of elementary operations that is : $\cap, \mathsf{Pre}_{>0}, \mathsf{Pre}_{>\delta}, \mathsf{Unreset}_R(,)$ and $\mathsf{Shrink}_{[-\delta,\delta]}$. The following lemma states that these elementary operator can be computed at the level of shrunk DBMs, in poly-time.

Lemma 24 ([10, 5]). *Let $M = f^0(N_1, \ldots, N_k)$ be an equation where f is an elementary operator and $N_1, \ldots, N_k$ are normalized DBMs. Let $P_1, \ldots, P_k$ be SMs. Then, there exists a SM Q such that $(M, Q) = f^\delta\big((N_1, P_1), \ldots, (N_k, P_k)\big)$. The SM Q and an upper bound on δ can be computed in poly-time.*

We now give some computational details about the result of Lemma 24 which we will need in subsequent proofs.

The operations on shrunk DBMs follow closely the counterparts on DBMs. Essentially, the matrices are defined on a richer algebra where elements are of the form $((m, \prec), p)$, which represent a constraint $x - y \prec m - \delta p$, instead of only $(m, \prec)$. We consider the following operations and relations. The addition is defined by $((m_1, \prec_1), p_1) + ((m_2, \prec_2), p_2) = ((m_1 + m_2, \prec), p_1 + p_2)$, where $\prec = \leq$ if $\prec = \prec' = \leq$, and $<$ otherwise.

Further, we let

$$((\alpha, \prec), k) \preceq ((\beta, \prec'), l) \quad \Leftrightarrow \quad \begin{cases} \alpha < \beta \text{ or} \\ \alpha = \beta \text{ and } k > l, \text{ or} \\ \alpha = \beta \text{ and } k = l \text{ and } \prec' = < \ \Rightarrow \ \prec = < . \end{cases} \tag{1}$$

Intuitively, the above relation can be obtained easily by instantiating δ to a sufficiently small value; the smaller constraint is the most restrictive. We also define the minimum as follows.

$$\min\left(((\alpha, \prec), k), ((\beta, \prec'), l)\right) = \begin{cases} ((\alpha, \prec), k) & \text{if } ((\alpha, \prec), k) \preceq ((\beta, \prec'), l), \\ ((\beta, \prec'), l) & \text{otherwise.} \end{cases} \tag{2}$$

Given these operations, elementary operations are computed in shrunk DBMs just like in usual DBMs. For instance, to compute $(O, R) = (M, P) \cap (N, Q)$, we let $(O, R)_{x,y} = \min\left((M, P)_{x,y}, (N, Q)_{x,y}\right)$, and we apply normalization (see below). To compute $\mathsf{Pre}\left((M, P)\right)$, we set the first row of the matrix to $((0, \leq), 0)$ and apply normalization.

It remains to explain the normalization procedure, which is again very similar to that in DBMs. For any DBM M, let $\mathsf{G}(M)$ denote the graph over nodes $\mathcal{C}_0$, where there is an edge $(x, y) \in \mathcal{C}_0^2$ of weight $M_{x,y}$ if $M_{x,y} < (\infty, <)$. The normalization of M corresponds to assigning to each edge (x, y) the weight of the shortest path in $\mathsf{G}(M)$. We say that M is *normalized* when it is stable under normalization [4].

Similarly to DBMs, the normalization of shrunk DBMs is defined using finite shortest paths in the graph $\mathsf{G}(M)$ of a given DBM M.

Definition 25. *Let M be a normalized DBM. For any $x, y \in \mathcal{C}_0$, we define $\Pi_{x,y}(\mathsf{G}(M))$ as the set of paths with least and finite M-weight from x to y in $\mathsf{G}(M)$.*

Notice that the shortest paths are defined with respect to weights in M and not M. In the rest of this paper, we consider paths $\pi = \pi_1 \ldots \pi_n \in \Pi_{x,y}(\mathsf{G}(M))$ *weighted by* a given DBM or any matrix. The M-sign of a path π, written $\mathsf{sign}_M(\pi)$ is $<$ if $\prec^M_{\pi_i, \pi_{i+1}} = <$ for some i, and $\leq$ otherwise. For a matrix P, the P-*weight* of π, written $P(\pi)$, (or the weight of π in P) is the sum of the weights $P_{\pi_j, \pi_{j+1}}$ for $1 \leq j \leq n - 1$. Finite-weighted shortest paths can be used to characterize the

non-emptiness and the normalization of shrunk DBMs. In fact, the normalization consists in replacing each component of a shrunk DBM with the weight of its shortest path.

Now, normalization consists in computing shortest paths in our algebra: Given a shrunk DBM (M, P) we define $\mathsf{norm}(M, P) = (M', P')$ as follows: for each $(x, y) \in \mathcal{C}_0$, let $P'_{x,y}$ be the largest P-weight of the paths in $\Pi_{x,y}(\mathsf{G}(M))$. We let $M'_{x,y} = (\mathsf{M}_{x,y}, \prec^{M'}_{x,y})$ where $\prec^{M'}_{x,y} = <$ if some path of $\Pi_{x,y}(\mathsf{G}(M))$ of P-weight $P'_{x,y}$ has sign $<$, and $\leq$ otherwise.

Lemma 26 ([5]). *Let M be a normalized non-empty DBM and P be a shrinking matrix. Then, (M, P) is non-empty if, and only if, for all $x \in \mathcal{C}_0$, there is no path in $\Pi_{x,x}(\mathsf{G}(M))$ with positive P-weight. Moreover, if (M, P) is not empty, then, writing $(M', P') = \mathsf{norm}(M, P)$, there exists $\delta_0 > 0$ such that $M' - \delta P'$ is normalized and defines the same set as $M - \delta P$ for all $\delta \in [0, \delta_0]$.*

C.2 Controllable Predecessors and Merge Operations

The goal of this subsection is to prove several properties about the controllable predecessor and merge operators. We prove that these operators yield non-empty sets along non-punctual paths. This crucial property will allow us to apply these operator to solve our robust timed games.

Let us first give the formal definition of CMerge on the whole unfolding.

Definition 27 (CMerge on trees). *Let t be a finite prefix of an unfolding $\mathsf{Unfold}(\mathcal{B}, q)$, denote by $q_1, \ldots, q_n$ the (labels of) leafs of t with $q_i = (\ell_i, r_i)$, and let $\delta \geq 0$. Let $(M_i)_{1 \leq i \leq n}$ be DBMs such that $M_i \subseteq r_i$, we define the set of δ-controllable predecessors of M_i's through t as follows:*

- *if the depth of t is equal to 0, then we have $n = 1$ and $q = q_1$ and we let: $\mathsf{CMerge}^{\delta}_t(M_1) = M_1$*
- *otherwise, we can write $t = q(t_1, \ldots, t_k)$. Then there exist indices i_j and n_j, with $1 \leq j \leq k$, such that (labels of) leafs of subtree t_j are precisely $q_{i_j}, \ldots, q_{i_j+n_j}$. We let q'_j denote the root of t_j, $e_j = q \to q'_j$, with $1 \leq j \leq k$, and define: $\mathsf{CMerge}^{\delta}_t((M_i)_{1 \leq i \leq n}) = \mathsf{CMerge}^{\delta}_{e_1,\ldots,e_k}((N_j)_{1 \leq j \leq k})$ where $N_j = \mathsf{CMerge}^{\delta}_{t_j}((M_p)_{i_j \leq p \leq i_j+n_j})$*

The following lemma shows that the controllable predecessors of a non-punctual path is non-empty provided that the target set has non-empty interior.

We consider the usual d_∞ metric on $\mathbb{R}^{\mathcal{C}}$, defined as $d_\infty(\nu, \nu') = \max_{x \in \mathcal{C}} |\nu(x) - \nu'(x)|$. We define $\mathsf{Ball}_{d_\infty}(u, \varepsilon)$ as the open ball of radius ε around the valuation u using the infinite norm on valuations.

Lemma 28 ([11]). *Let π be a non-punctual path from a region r to s. Let $s' \subseteq s$ such that there exists $v' \in s'$ and $\varepsilon > 0$ with $\mathsf{Ball}_{d_\infty}(v', \varepsilon) \cap s \subseteq s'$. Then, $\mathsf{CPre}^{\delta}_{\pi}(s')$ is non-empty for small enough $\delta > 0$.*

We state here a more precise version of the above lemma.

Lemma 29. *Let π be a non-punctual path from a region r to s. Let $r' \subseteq r$ and $s' \subseteq s$ such that there is $v' \in s'$ and $\varepsilon > 0$ with $\mathsf{Ball}_{d_\infty}(v', \varepsilon) \cap s \subseteq s'$, and $r' = \mathsf{CPre}^0_\pi(s')$. We suppose that r' and s' are defined by DBMs. Consider P a SM such that $(s', P) \neq \emptyset$. Then, there exists Q a SM such that:*

- *$(r', Q) = \mathsf{CPre}^\delta_\pi(s', P)$, and*
- *(r', Q) is non-empty for small enough δ.*

Proof. To prove the first item, it is sufficient to notice that CPre is an elementary operator, thus thanks to Lemma 24 there exists a SM Q such that the first item holds.

Let us show the second item. For this notice first that $(s', P) \neq \emptyset$ means that for small enough $\delta > 0$, $s' - \delta \cdot P$ has a non-empty interior in the topology induced on s. Therefore, the requirements of Lemma 28 hold, and it follows that there exists δ' such that $\mathsf{CPre}^{\delta'}_\pi(s' - \delta \cdot P) \neq \emptyset$. To obtain the desired result, it is sufficient to show that the second item holds when $\delta' = \delta$. Assume first that $\delta' \geq \delta$, then the second item of the lemma follows because $\mathsf{CPre}^{\delta_0}_\pi(Z) \subseteq \mathsf{CPre}^{\delta_1}_\pi(Z)$ for $\delta_0 > \delta_1$. If $\delta' < \delta$, then the result still holds because $z - \delta \cdot M \subseteq z - \delta' \cdot M$ for $\delta' < \delta$. □

A similar property can be proved on the CMerge operator.

Lemma 30. *Let $r, r_1, \cdots, r_n$ be a set of regions satisfying $r = \mathsf{CMerge}^0_{e_1,\ldots,e_n}(r_1, \cdots, r_n)$. Consider $r' \subseteq r$, and $r'_i \subseteq r_i$ for each i such that there is $v'_i \in r'_i$ and $\varepsilon > 0$ with $\mathsf{Ball}_{d_\infty}(v'_i, \varepsilon) \cap r_i \subseteq r'_i$, and $r' = \mathsf{CMerge}^0_{e_1,\ldots,e_n}(r'_1, \cdots, r'_n)$. We suppose that $r', r'_1, \ldots, r'_n$ are defined by DBMs. Let $P_1, \cdots, P_n$ be a set of SMs such that $(r'_i, P_i) \neq \emptyset$. Then there exits a SM Q such that:*

- *$(r', Q) = \mathsf{CMerge}^\delta_{e_1,\ldots,e_n}((r'_i, P_i)_i)$, and*
- *(r', Q) is non-empty for small enough δ.*

Proof. Let us denote by $(\ell, g_i, a, R_i, \ell_i)$ the transition associated with edge e_i, and denote by s the non-punctual intermediate region time-successor of r and such that $s[R_i \leftarrow 0] = r_i$. For any i, there exists a SM P'_i such that $\mathsf{Unreset}_{R_i}((r'_i, P_i)) \cap s = (s'_i, P'_i) \neq \emptyset$ where $s'_i = \mathsf{Unreset}_{R_i}(r'_i) \cap s$. Moreover because (r'_i, P_i) has non-empty interior in the topology induced on r_i and s is non-punctual, s'_i is also non-punctual. Then there exists a SM P' such that $\bigcap_{i=1}^n (s'_i, P'_i) = (s', P') \neq \emptyset$, where s' is non-punctual. We get the desired result thanks to the following facts. On one hand, r is a temporal predecessor of s'. Therefore, there exists a valuation $\nu \in r$ and a delay d such that we have $\nu \cdot d = \nu'$. On the other hand, s' is non-punctual, hence there exists $\eta > 0$ such that $\nu' + [-\eta, \eta] \subseteq \mathsf{Ball}_{d_\infty}(\nu', \varepsilon')$. □

It is then easy to deduce from Lemmas 29 and 30 that Lemma 10 holds, by induction on the size of the (finite) prefix of an unfolding.

C.3 The notion of 0-dominance

The subsection contains the proof of the results related to 0-dominance. The goal of these developments is to prove Lemma 11, which is the only lemma used in the rest of the proof.

The following lemma shows that the 0-dominance relation is monotonous; in other terms, it shows how the zeros of the shrinking matrices are (backward-) propagated along computations.

Lemma 31. *The operations* $\mathsf{Pre}, \mathsf{Unreset}(), \cap$ *are monotonous for the relation* $\preceq_0$ *in the following sense.*

- *Let* $(N, P') = (M_1, P_1) \cap (M_2, P_2)$ *and* $(N, Q') = (M_1, Q_1) \cap (M_2, Q_2)$. *Then* $P_1 \preceq_0 Q_1 \wedge P_2 \preceq_0 Q_2 \Rightarrow P' \preceq_0 Q'$.
- *Let* $(N, P') = \mathsf{Pre}((M, P))$ *and* $(N, Q') = \mathsf{Pre}((M, Q))$. *Then* $P \preceq_0 Q \Rightarrow P' \preceq_0 Q'$.
- *Let* $(N, P') = \mathsf{Unreset}_R((M, P))$ *and* $(N, Q') = \mathsf{Unreset}_R((M, Q))$. *Then* $P \preceq_0 Q \Rightarrow P' \preceq_0 Q'$.

Proof.
- That $Q' \leq P'$ follows from the monotonicity of the operation $(Z, Z') \mapsto Z \cap Z'$. We show that $Q'_{x,y} = 0$ implies $P'_{x,y} = 0$ for any x, y. We recall the intersection operation on shrunk DBMs. We first define a SM P'' as follows. For any $x, y \in \mathcal{C}_0$, if $(M_i)_{x,y} < (M_{3-i})_{x,y}$ for some $i = 1, 2$, then $(N'_i)_{x,y} = (M_i)_{x,y}$ and $P''_{x,y} = (P_1)_{x,y}$. Otherwise $P''_{x,y} = \max((P_1)_{x,y}, (P_2)_{x,y})$. Then, (N, P') is the normalization of (N', P''). Let us denote by Q'' this intermediate SM computed for the other equation. It is clear that under the assumption $P_1 \preceq_0 Q_1 \wedge P_2 \preceq_0 Q_2$, $Q''_{x,y} = 0$ implies $P''_{x,y} = 0$. Now, the normalization consists in assigning to each $P'_{x,y}$, the maximum P''-weight of the shortest paths from x to y in $\mathsf{G}(N')$. So, if $P'_{x,y} > 0$, then there exists a shortest path in $\mathsf{G}(N')$ from x to y whose P''-weight is positive, that is, there exists an edge (z, z') in this path with $P''_{z,z'} > 0$. But we must have $Q''_{z,z'} > 0$ by assumption. This implies that $Q'_{x,y} > 0$.
- That $Q' \leq P'$ follows again from the monotonicity of the operation $Z \mapsto \mathsf{Pre}(Z)$. Given (M, P), the $\mathsf{Pre}((M, P))$ is computed as follows. We first define (M', P'') by assigning $(0, 0)$ to all components of the first row, and then define (N, P') as the normalization of (M', P''). By the same reasoning on the normalization procedure as in the first case, we obtain that $P \preceq_0 Q$ implies $P' \preceq_0 Q'$.
- That $Q' \leq P'$ follows again from the monotonicity of the operation $Z \mapsto \mathsf{Unreset}_R(Z)$. The unreset operation is defined as $\mathsf{Unreset}_R(Z) = \mathsf{free}_R(R = 0 \wedge Z)$, that is we first intersect with the zone $\wedge_{x \in R} x = 0$, and free all constraints on the clocks $x \in R$. The intersection was treated by the first case of this lemma. Thus, if we denote $(M', P_1) = (R = 0) \wedge (M, P)$ and $(M', Q_1) = (R = 0) \wedge (M, Q)$, then we know that $P_1 \preceq_0 Q_1$. We now concentrate on the free_R operation which consists in replacing by $(\infty, 0)$ all components (x, y) of the given shrunk DBM where $x \in R$ or $y \in R$, and applying normalization. The 0-dominance relation is preserved before normalization by construction, and after normalization by the same reasoning as in the first case of this lemma.

□

The following two lemmas compare the operators CPre and CMerge with respect to 0-dominance.

Lemma 32. *Let* r, r', r_1, r_2 *be regions and* P, P_1, P_2, Q, Q_1, Q_2 *be SMs such that:*

- $(r, P) = \mathsf{CMerge}^0_{a,r'}((r_1, P_1), (r_2, P_2)) \cap r$,
- $(r, Q) = \mathsf{CPre}^0_{e_1}(r_1, Q_1) \cap \mathsf{CPre}^0_{e_2}(r_2, Q_2) \cap r$,
- $P_i \preceq_0 Q_i$ *for* $i \in \{1, 2\}$.

Then $P \preceq_0 Q$.

Proof. We start by rewriting the shrunk DBMs under consideration. We have :

$$\begin{aligned}(r,P) &= \mathsf{CMerge}^0_{a,r'}((r_1,P_1),(r_2,P_2)) \cap r \\ &= \mathsf{Pre}(r' \cap \mathsf{Unreset}_{R_1}((r_1,P_1)) \cap \mathsf{Unreset}_{R_2}((r_2,P_2))) \cap r \\ &= \mathsf{Pre}((r' \cap \mathsf{Unreset}_{R_1}((r_1,P_1))) \cap (r' \cap \mathsf{Unreset}_{R_2}((r_2,P_2)))) \cap r \ .\end{aligned}$$

and:

$$\begin{aligned}(r,Q) &= \mathsf{CPre}^0_{e_1}(r_1,Q_1) \cap \mathsf{CPre}^0_{e_2}(r_2,Q_2) \cap r \\ &= \mathsf{Pre}(r' \cap \mathsf{Unreset}_{R_1}((r_1,Q_1))) \cap \mathsf{Pre}(r' \cap \mathsf{Unreset}_{R_2}((r_2,Q_2))) \cap r \ .\end{aligned}$$

We let $(r', P'_i) = r' \cap \mathsf{Unreset}_{R_i}((r_i,P_i))$, $(r', Q'_i) = r' \cap \mathsf{Unreset}_{R_i}((r_i,Q_i))$, and $(r, Q''_i) = r \cap \mathsf{Pre}((r',Q'_i))$ for $i \in \{1,2\}$ and we obtain :

$$\begin{aligned}(r,P) &= \mathsf{Pre}((r', \max(P'_1,P'_2))) \ , \\ (r,Q) &= (r, \max(Q''_1,Q''_2)) \ .\end{aligned}$$

We let $(M,P_M) = \mathsf{Pre}\left((r',P'_1) \cap (r',P'_2)\right)$ and $(M,Q_M) = \mathsf{Pre}\left((r',Q'_1)\right) \cap \mathsf{Pre}\left((r',Q'_2)\right)$. It suffices to show that $P_M \preceq_0 Q_M$ since this implies, by Lemma 31 that $P \preceq_0 Q$. Again by Lemma 31, we already know that $P'_i \preceq_0 Q'_i$ for all $i = 1,2$.

We will need the following observation. For any shrunk DBMs satisfying $(N',R') = \mathsf{Pre}((N,R))$, for any $x,y \in \mathcal{C}_0$ with $x \neq 0$, $R_{x,y} = R'_{x,y}$. In other terms, the diagonal constraints and upper bounds are preserved under the pretime operation. This can be seen as follows. In DBMs, the pretime consists in assigning 0 to the first row and applying normalization. Because the first row always takes nonpositive values, the normalization only changes the first row. In fact, consider any component (z,z') with $z \neq 0$. Because the value of a component $(0,x)$ cannot decrease after being assigned 0, this operation does not introduce any new shortest paths from z to z', so the component (z,z') is not updated during normalization. In shrunk DBMs, the situation is similar: because no new shortest paths are introduced from z to z', $R_{z,z'}$ is not updated during normalization, hence $R'_{z,z'} = R_{z,z'}$.

We now show $P_M \preceq_0 Q_M$. Assume that $(P_M)_{x,y} > 0$ for some $x,y \in \mathcal{C}_0$ with $y \neq 0$. Given the previous remark, we get that either $(P'_1)_{x,y} > 0$ or $(P'_2)_{x,y} > 0$. But because $P'_i \preceq_0 Q'_i$, this means that either $(Q'_1)_{x,y} > 0$ or $(Q'_2)_{x,y} > 0$. It follows that $\mathsf{Pre}((r',Q'_i))_{x,y} > 0$ for some $i = 1,2$, thus $(Q_M)_{x,y} > 0$. It remains to show that $(P_M)_{0,x} > 0$ implies $(Q_M)_{0,x} > 0$. Let $(r',P'') = (r',P'_1) \cap (r',P'_2)$. Because the pretime operation first resets the row of (r',P'') to $(0,0)$ before applying normalization, this means that there exists a shortest path in $\mathsf{G}(M)$ from 0 to x along which there is an edge (z,z') such that $(P'')_{z,z'} > 0$. This implies that $(P'_i)_{z,z'} > 0$ for some $i = 1,2$, so $(Q'_i)_{z,z'} > 0$. It follows that the component (z,z') of $\mathsf{Pre}((r',Q'_i))$ is positive, so $(Q_M)_{z,z'} > 0$.

□

Lemma 33. *Let r, r_1, r_2 be regions and P, P_1, P_2, Q, Q_1, Q_2 be SMs such that:*

- $(r,P) = \mathsf{CPre}^0_{\pi_0}\left(\mathsf{CPre}^0_{\pi_1}((r_1,P_1)) \cap \mathsf{CPre}^0_{\pi_2}(r_2,P_2))\right)$,
- $(r,Q) = \mathsf{CPre}^0_{\pi_0\pi_1}(r_1,Q_1) \cap \mathsf{CPre}^0_{\pi_0\pi_2}(r_2,Q_2)$,

– $P_i \preceq_0 Q_i$ *for* $i \in \{1,2\}$.

Then $P \preceq_0 Q$.

Proof. Let us write $(s, P'_i) = \mathsf{CPre}^0_{\pi_1}((r_i, P_i))$ and $(s, Q'_i) = \mathsf{CPre}^0_{\pi_1}((r_i, Q_i))$ for $i = 1, 2$. We rewrite

– $(r, P) = \mathsf{CPre}^0_{\pi_0}((s, P'_1) \cap (s, P'_2))$,
– and $(r, Q) = \mathsf{CPre}^0_{\pi_0}((s, Q'_1)) \cap \mathsf{CPre}^0_{\pi_0}((s, Q'_2))$.

By Lemma 31, we have $P'_i \preceq_0 Q'_i$. We prove the result by induction on $|\pi_0| \geq 0$.

For the base case, assume that $\pi_0 = e$ consists of one edge. In this case, notice that $(r, P) = \mathsf{CMerge}_{a,r'}((s, P'_1), (s, P'_2))$. The result then follows from Lemma 32.

Assume $|\pi_0| > 0$ and write $\pi_0 = e\pi'$. Let us write

– $(s, P^1) = \mathsf{CPre}^0_{\pi'}((s, P'_1) \cap (s, P'_2))$,
– $(s, Q^1) = \mathsf{CPre}^0_{\pi'}((s, Q'_1)) \cap \mathsf{CPre}^0_{\pi'}((s, Q'_2))$.

By induction hypothesis, we have $P^1 \preceq_0 Q^1$. We now consider

– $(r, P) = \mathsf{CPre}^0_e\left(\mathsf{CPre}^0_{\pi'}((s, P'_1) \cap (s, P'_2))\right)$,
– $(r, Q^2) = \mathsf{CPre}^0_e\left(\mathsf{CPre}^0_{\pi'}((s, Q'_1)) \cap \mathsf{CPre}^0_{\pi'}((s, Q'_2))\right)$.

which yields $P \preceq_0 Q^2$ by Lemma 31. Last, we consider the following equations.

– $(r, Q^2) = \mathsf{CPre}^0_e\left(\mathsf{CPre}^0_{\pi'}((s, Q'_1)) \cap \mathsf{CPre}^0_{\pi'}((s, Q'_2))\right)$.
– $(r, Q) = \mathsf{CPre}^0_e\left(\mathsf{CPre}^0_{\pi'}((s, Q'_1))\right) \cap \mathsf{CPre}^0_e\left(\mathsf{CPre}^0_{\pi'}((s, Q'_2))\right)$,

By Lemma 32, applied with $e_1 = e_2$, we get that $Q^2 \preceq_0 Q$. It follows that $P \preceq_0 Q$. □

D EXPTIME-hardness

We reduce the halting problem for linearly-bounded alternating Turing machines to the reachability of an absorbing state in our semantics, which implies the hardness for Büchi objectives as well. We first explain the reduction for robust timed games, i.e. in the adversarial semantics. At the end of this section, we will adapt the reduction to almost-sure reachability in the MDP semantics. Note that the EXPTIME-hardness of the stochastic game semantics follows from the equivalence with the adversarial semantics.

D.1 Hardness of Robust Timed Games

The reduction follows that of [11] from non-alternating Turing machines that shows the PSPACE-hardness of the problem when Perturbator can only perturb delays. The encoding is quite standard for timed automata with the particularity that it should allow the operations to be defined without the use of equality guards. The alternation is introduced thanks to the ability of both players to determine the successor locations.

An alternating Turing machine contains three types of states:

– (disjunction) $\mathsf{trans}(q) = q' \vee q''$
– (conjunction) $\mathsf{trans}(q) = q' \wedge q''$
– (instruction) $\mathsf{trans}(q) = (\gamma, \gamma', \mathsf{dir}, q')$ where $\gamma, \gamma' \in \{a, b\}$ and $\mathsf{dir} \in \{\leftarrow, \rightarrow\}$. Such a transition reads a γ in the current cell, write a γ' and follows direction given by dir.

We do not recall the acceptance condition on disjunctive and conjunctive states, and refer to [7]. We assume that there is one distinguished state halt where the machine halts.

Let $\mathcal{M}$ be a linearly-bounded alternating Turing machine, and w_0 be an input to $\mathcal{M}$. Let N be the bound on the tape of $\mathcal{M}$ when simulating on input word w_0 (N is linear in $|w_0|$). We assume the alphabet is $\{a, b\}$, and we encode the content of the i-th cell C_i of $\mathcal{M}$ using a clock x_i with the following convention: when the module is entered, cell C_i contains an a whenever $x_i < 1$, and it contains a b whenever $x_i > 2$.

To simulate a transition at a state q with $\mathsf{trans}(q) = (\alpha, \beta, \mathsf{dir}, q')$ of $\mathcal{M}$, where $\alpha, \beta \in \{a, b\}$ and $\mathsf{dir} \in \{-1, 1\}$, we build a module as in Fig. 9 for every index i such that both i and $i + \mathsf{dir}$ lie between 1 and N. Along this module, after any initial delay of duration $u_0 \in (2, 3)$, cell C_j contains a iff $x_j < 4$, and it contains b iff $x_j > 4$. Transitions between p_j and p_{j+1}, for $j \neq i$, ensure preservation of this encoding. Between p_i and p_{i+1}, the module checks that $C_i = \alpha$ (through guard $g_{\alpha,i}$), and replace this content with β (through reset $Y_{\beta,i}$). This way, one run through the module updates the content of the tape and the position of the tape head according to the selected transition of $\mathcal{M}$.

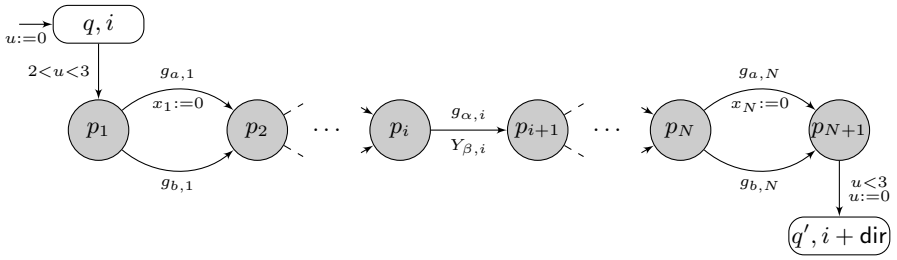

Fig. 9. Simulating states q with $\mathsf{trans}(q) = (\alpha, \beta, \mathsf{dir}, q')$ where $\alpha, \beta \in \{a, b\}$. Index i is such that $1 \leq i \leq N$ and $1 \leq i + \mathsf{dir} \leq N$. Guards and resets are defined as follows: $g_{a,j}$ is $(x_j < 4 \wedge u < 3)$ and $g_{b,j}$ is $(x_j > 4 \wedge u < 3)$, while $Y_{a,j}$ is $\{x_j\}$ and $Y_{b,j}$ is empty. Notice that at any state p_i, only one transition is enabled. So we assume that all edges have distinct labels in this module.

A disjunctive state q with $\mathsf{trans}(q) = q' \vee q''$ is simulated by the following module, which allows $\mathsf{Controller}$ to pick any successor.

A conjunctive state q with $\mathsf{trans}(q) = q' \wedge q''$ is simulated by the following module, which allows $\mathsf{Perturbator}$ to pick the successor.

We also need a simple normalization module which will make sure that the clocks are bounded. Assume that an a in cell C_j is encoded with clock constraint $x_j < 1$, and a b with clock constraint $2 < x_j < 4$ when entering a state (q, i). Then all above modules ensure that, when leaving the module, either $x_j < 1$ (in

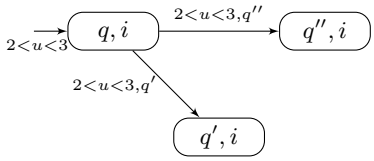

Fig. 10. Simulating states q with $\mathsf{trans}(q) = q' \vee q''$. The edges leaving the state (q, i) have distinct labels q' and q'' which correspond to Controller's choice.

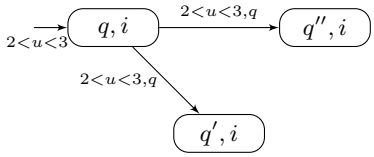

Fig. 11. Simulating states q with $\mathsf{trans}(q) = q' \wedge q''$. The edges leaving the state (q, i) have the same label q. Thus Perturbator has full control on the successor location at this state.

Fig. 9, in case cell C_j now contains an a), or either $2 < x_j < 4$ or $4 < x_j < 7$ (the two possible cases for cell C_j to contain a b). It is easy to plug at the end of each module another module, which ensures that $2 < x_j < 4$ when cell C_j contains an a, and $x_j < 1$ when cell C_j contains a b. Then we can plug at the end of this second module another one which ensures that $x_j < 1$ when cell C_j contains an a, and $2 < x_j < 4$ when cell C_j contains a b, which is the initial encoding. Using this trick, the constructed timed automata has only bounded clocks.

We complete the construction with an initialization module (encoding input w_0 on the tape). We write $\mathcal{A}$ for this timed automaton and write halt for the halting location (which is made a sink). The set $\{\mathsf{halt}\}$ is the Büchi condition.

We show the following lemma:

Lemma 34. *Let s_0 be the initial configuration of $\mathcal{A}$. The following conditions are equivalent:*

1. *$\mathcal{M}$ has a halting computation on w_0;*
2. *Controller has a strategy in $\mathcal{G}_\delta(\mathcal{A})$ to reach halt, for all $\delta < \frac{1}{4N}$;*

Proof. First assume that $\mathcal{M}$ has a halting computation on w_0. We describe a robust winning strategy for the Controller in $\mathcal{A}$. We consider the simulation of the state q and transition $(\alpha, \beta, \mathsf{dir}, q')$ of $\mathcal{M}$. Assume state (q, i) is entered with some valuation v_0 that properly encodes some content of the tape of $\mathcal{M}$: if cell C_j contains an a, then $v_0(x_j) < 1$, and if cell C_j contains a b, then $v_0(x_j) > 2$. We describe a strategy for Controller, and then pick a corresponding outcome (with perturbations). The candidate strategy is as follows: leave (q, i) when $u = 2.5$, and in each of the p_j's, wait for δ time units, and then takes the next feasible transition.

Under this strategy, and assuming that Perturbator perturbs by $-\delta \leq \varepsilon_0 \leq \delta$ on transition leaving (q, i), and by $-\delta \leq \varepsilon_j \leq \delta$ on transition leaving p_j, the valuation when arriving in p_j is v_j (resp. in $(q', i + \mathsf{dir})$ is v'_0) such that:

– if $i \neq j$ and cell C_j contains a b, or if $i = j$ and $\beta = b$, then for every $1 \leq k \leq N+1$, $v_k(x_j) = v_0(x_j) + 2.5 + (k-1)\delta + \sum_{h=0}^{k-1} \varepsilon_h$;
– if $i \neq j$ and cell C_j contains an a, or if $i = j$ and $\beta = a$, then for every $1 \leq k \leq j$, $v_k(x_j) = v_0(x_j) + 2.5 + (k-1)\delta + \sum_{h=0}^{k-1} \varepsilon_h$, $v_{j+1}(x_j) = 0$, and for every $j+2 \leq k \leq N+1$, $v_k(x_j) = (k-j-1)\delta + \sum_{h=j+1}^{k-1} \varepsilon_h$.

Note that in all cases, since $\delta < \frac{1}{4N}$, if cell C_j contains an a, then $v_j + \delta + \varepsilon_j \models g_{a,j}$, and if cell C_j contains a b, then $v_j + \delta + \varepsilon_j \models g_{b,j}$. Note also that if either $i \neq j$ and cell C_j contains an a, or if $i = j$ and $\beta = a$, then $v_0'(x_j) < 1$, and if either $i \neq j$ and cell C_j contains a b, or if $i = j$ and $\beta = b$, then $v_0'(x_j) > 2$. This means that whatever the perturbation, at the end of the module the clocks properly encode the new content of the tape of $\mathcal{M}$ after the mentioned transition has been taken. The same strategy is applied in the normalization modules; this ensures that perturbations are not accumulated too much along disjunctive or conjunctive states.

We can apply this strategy in every module simulating a transition of the Turing machine. Since $\mathcal{M}$ halts on w_0, the halt state can be reached by following this strategy.

Conversely, if $\mathcal{M}$ does not halt on w_0, then when playing without perturbations, Controller cannot reach the halting state. In particular, Controller has no robust strategy. □

D.2 Hardness of the MDP semantics

We consider the same reduction with a slight modification. Observe that if a linearly bounded alternating Turing machine M accepts a word w_0, then it accepts in at most $|M|2^N$ steps. In other terms, this is the bound on the execution tree witnessing the acceptance.

Now, in each module we constructed, the automaton spends at most $3 + 4N$ time units, where 3 time units are spent for simulating a transition, and at most $4N$ for normalization of the encoding. Given an instance M, w_0, let $\mathcal{A}'$ denote the timed automaton obtained from $\mathcal{A}$ by adding a new clock t which is guarded by $t \leq (3+4N)|M|2^N$ at each transition, except on the self-loop of state halt. We can now rewrite Lemma 34, as follow: the word w_0 is accepted by M if, and only if Controller has a strategy in $\mathcal{G}_\delta(\mathcal{A}')$ reaching halt for $\delta < \frac{1}{4N}$. Note also that $\mathcal{A}'$ can be constructed in polynomial time. In particular, the constants can be encoded in polynomial space.

Because time progresses in each module of $\mathcal{A}'$ the only infinite behaviors are those reaching halt. The reduction for the MDP semantics follows. In fact, if the Turing machine accepts, then Controllerhas a strategy ensuring that all compatible runs reach halt in less than $(3+4N)|M|2^N$ time units, so this also holds if perturbations and the non-determinism are probabilistic. Conversely, if M does not accept, then for any Controllerstrategy, there exists a choice for the conjunctions such that the machine does not end in halt after $|M|2^N$ steps; it follows that with positive probability, halt is not reached before $(3+4N)|M|2^N$ time units.

Lemma 35. *Let s_0 be the initial configuration of $\mathcal{A}$. The following conditions are equivalent:*

1. $\mathcal{M}$ *has a halting computation on* w_0*;*
2. Controller *has a strategy in* $\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A}')$ *to reach* halt *almost surely, for all* $\delta < \frac{1}{4N}$*;*

E Probabilistic Semantics

In this section, we formally define a probability space on timed automata runs under given strategies, and prove our results in this setting. Our definitions follow closely [2]. In particular, we define the probability measure on cylinders defined by the same *constrained symbolic paths* (see below). This allows us to reuse the proofs of [2] to derive the well-definedness of our measure.

We start by formally defining the stochastic game semantics, and then define the MDP semantics as a particular case.

E.1 Probability Measures

Fix a timed automaton $\mathcal{A}$. For any state $s = (\ell, \nu)$, and set of edges $e_1 \dots e_n$, we will define a symbolic path as the set of runs that start at s and follow the edges $e_1 \dots e_n$. We will moreover consider a constraint $\mathcal{C} \subseteq \mathbb{R}^n$ over the sequence of delays of such runs. Formally,

$$\pi_{\mathcal{C}}(s, e_1 \dots e_n) = \{\rho = s \xrightarrow{\tau_1, e_1} s_1 \to \dots \to s_n \mid \rho \in \mathsf{Runs}_f(\mathcal{A}, s), (\tau_i)_i \in \mathcal{C}\}.$$

We define the *cylinder* of $\pi_{\mathcal{C}}(s, e_1 \dots e_n)$ as the set of infinite runs whose prefix of length n belongs to $\pi_{\mathcal{C}}(s, e_1 \dots e_n)$.

Given a pair of strategies, we will define a probability measure based on these cylinders. A Controller's strategy σ is *measurable* if for any action $a \in \mathsf{Act}$, the subset of histories $\sigma^{-1}(\mathbb{R} \times \{a\})$ is Lebesgue-measurable. A Perturbator's strategy τ is *measurable* if for any edge e, the subset of histories $\tau^{-1}(\{e\})$ is Lebesgue-measurable.

We will write the histories as sequences of the form $sd_1t_1e_1 \dots d_nt_ne_n$, where s is the starting state, d_i the delay suggested by Controller, t_i the perturbed delay (such that $|t_i - d_i| \leq \delta$, and e_i the edge that is taken after the delay. Histories where Controller has made a move will be denoted by $sd_1t_1e_1 \dots d_nt_ne_n(d_{n+1}, a_{n+1})$. Note that e_i determines the action chosen by Controller along with the delay d_i. For Controller's strategy σ, we denote by $\sigma_d(h)$ the delay prescribed at history h.

In the probabilistic setting, we will restrict to pairs of measurable strategies. Let σ, τ be such a pair. Consider a sequence of edges $\bar{e} = e_1 \dots e_n$, and denote by a_i the label of edge e_i. We consider a sequence of nonnegative reals $t_1, \dots, t_n$, and use the notation $\bar{t}^i = (t_1, \dots, t_i)$. We also denote by $h_{s,i}^{\sigma,\tau} = sd_1t_1e_1 \dots d_it_ie_i$, the history of length i starting at s under σ, τ. We define for $1 \leq i < n$,

$$\begin{aligned} I_i^{\sigma,\tau}(s, \bar{t}^{i-1}, \bar{e}) = \sigma_d(h_{s,i-1}^{\sigma,\tau}) \pm \delta \\ \cap \{t_i \mid h_{s,i}^{\sigma,\tau} \in \sigma^{-1}(\mathbb{R} \times \{a_{i+1}\})\} \\ \cap \{t_i \mid h_{s,i}^{\sigma,\tau} \cdot (d_i, a_i) \in \tau^{-1}(\{e_{i+1}\})\} \end{aligned}$$

For $i = n$, we let $I_n^{\sigma,\tau}(s, \bar{t}^n, \bar{e}) = \sigma_d(h_{s,n-1}^{\sigma,\tau}) \pm \delta$. This defines the set of perturbed delays that can be observed at given history when Controller has strategy σ, such

that the players' next moves conform to the edge sequence $\bar{e}$. By hypothesis, each set $I_i^{\sigma,\tau}(\cdot)$ is measurable. [7] For any measurable constraint $\mathcal{C}$, we define the probability of a constrained symbolic path as follows.

$$\mathbb{P}^{\sigma,\tau}_{\mathcal{G}_\delta^{\mathsf{SG}}(\mathcal{A}),s}(\pi_{\mathcal{C}}(s, e_1 \cdots e_n)) = \\ \int_{t_1 \in I_1^{\sigma,\tau}(s,\bar{e})} \int_{t_2 \in I_2^{\sigma,\tau}(s,\bar{t}^1,\bar{e})} \int_{t_n \in I_n^{\sigma,\tau}(s,\bar{t}^{n-1},\bar{e})} \mathbf{1}_{\mathcal{C}} \frac{dt_1}{2\delta} \cdot \frac{dt_2}{2\delta} \cdot \ldots \cdot \frac{dt_n}{2\delta},$$

We initialize by $\mathbb{P}^{\sigma,\tau}_{\mathcal{G}_\delta^{\mathsf{SG}}(\mathcal{A}),s}(\pi_{\mathcal{C}}(s)) = 1$. This definition makes sure that we sum over values for $t_1, \ldots, t_n$ which are compatible with the strategies, satisfies $\mathcal{C}$, and follow the symbolic path $e_1 \ldots e_n$. Note that each integral is well-defined thanks to the measurability assumption on the strategies.

We define the MDP semantics by restricting to a unique strategy of Perturbator, consisting, given a move (d, a) by Controller, in choosing all available edges with label a with equal probability. We define the probability measure $\mathbb{P}^{\sigma}_{\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A}),s}(\cdot)$ given state s, and Controller strategy σ.

Let us denote $\Omega^s_{\mathcal{A}}$ the σ-algebra generated by the cylinders.

Theorem 36. *For any timed automaton $\mathcal{A}$, state s, and pair of measurable strategies σ, τ, $\mathbb{P}^{\sigma,\tau}_{\mathcal{G}_\delta^{\mathsf{SG}}(\mathcal{A}),s}$ and $\mathbb{P}^{\sigma}_{\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A}),s}$ define a probability measure over $(\mathsf{Runs}(\mathcal{A}, s), \Omega^s_{\mathcal{A}})$.*

Proof. The proof follows step by step that of [2], see [3, pages 40-42]. □

Notice that the probability measure in the MDP semantics is a particular case of that of the stochastic game semantics; so it suffices to show the correctness of the latter.

Note that the set of Büchi-accepting runs is measurable since it can be written as the intersection, for $n > 0$, of all cylinders that visit at least n times accepting locations. Let us denote by ϕ the set of runs that are accepting for the given Büchi condition. We say that a strategy σ for Controller is *almost surely winning* if for any Perturbator's strategy τ, $\mathbb{P}^{\sigma,\tau}_{\mathcal{G}_\delta^{\mathsf{SG}}(\mathcal{A}),s}(\phi) = 1$.

E.2 Stochastic Game Semantics: Characterization

Our main result on the stochastic game semantics is that considering probabilistic perturbations rather than adversarial ones do not weaken Perturbator: the characterization for the robust game semantics also holds in the stochastic game semantics. The idea is the following. If condition $\mathcal{C}_C$ holds, then we know that Controller can win against adversarial perturbations, so he can also win against Perturbator with probabilistic perturbations. The converse is more tricky: we show that if $\mathcal{C}_P$ holds, then under any Controllerstrategy, a non-aperiodic cycle is repeated a large number of times, and almost surely the run is blocking; thus finite, and fails to satisfy the Büchi condition.

We will now formalize these ideas and give the full proof. Let us first recall some definitions: We define ε-far runs and show that such runs are blocking along non-aperiodic cycles.

[7] In fact, if $\mathsf{Cyl}(h^{\sigma,\tau}_{s,i-1})$ denotes all histories extending $h^{\sigma,\tau}_{s,i-1}$, then we have $I_i^{\sigma,\tau}(\cdot) = p_i\left(\mathsf{Cyl}(h^{\sigma,\tau}_{s,i-1}) \cap \sigma^{-1}(\mathbb{R} \times \{a_{i+1}\})\right)$, where p_i denotes the canonical projection from $\mathbb{R}^n$ to the i-th component.

Let us call a valuation v *ε-far* if $v + [-\varepsilon, \varepsilon] \subseteq [v]$. A run is ε-far if all delays end in ε-far valuations. We show that after any Controller's move the perturbed delay is ε-far with nonzero probability.

Lemma 37. *Given any $\delta > 0$, and any timed automaton with C clocks, let $\varepsilon = \frac{\delta}{4(C+1)}$. For any state (ℓ, ν), and* Controller*'s valid move (d, a), there exists an interval $I \subseteq [d - \delta, d + \delta]$ such that $|I| \geq \varepsilon$, and for all $d' \in I$, $\nu + d'$ is ε-far.*

Proof. After any delay $\nu \xrightarrow{d} \nu'$, $d \geq \delta$, chosen by Controller, consider the regions spanned by the set $\nu' + [0, \delta]$. It is easy to see that this set intersects at most $|\mathcal{C}| + 1$ different regions (See also [5, Lemma 6]), all of which must satisfy the guard by definition of the game. So some region r satisfies $\nu' + [\alpha, \beta] \subseteq r$, for some $0 \leq \alpha < \beta \leq \delta$ with $\beta - \alpha \geq \frac{\delta}{|\mathcal{C}|+1}$. Thus, $[\alpha, \beta]$ contains an interval with desired properties. □

We say that a run is ε-far if all perturbed delays end in ε-far valuations. It was shown in [11] that these runs along non-aperiodic cycles necessarily lead to a deadlock.

Lemma 38 ([11, Section 5]). *For any timed automaton $\mathcal{A}$, let ε be as in Lemma 37. There exists $n > 0$ such that along any non-aperiodic cycle, there is no ε-far run of length n.*

Before we prove Theorem 13, observe that the strategy σ derived from condition $\mathcal{C}_C$ is measurable since it has finite-memory and described via shrunk DBMs. Similarly, the strategy τ derived from $\mathcal{C}_P$ is also measurable.

Now the idea behind the proof in our case is to show that along any infinite run, almost-surely, some n consecutive delays will be ε-far.

Theorem 13. *It is EXPTIME-complete to decide whether for some $\delta > 0$,* Controller *has a strategy achieving a given Büchi objective almost surely in the stochastic game semantics. Moreover, if $\mathcal{C}_C$ holds then* Controller *wins almost-surely; if $\mathcal{C}_P$ holds then* Perturbator *wins almost-surely.*

Proof. We first show that if $\mathcal{C}_C$ holds then Controller wins almost-surely. This is a straightforward corollary of the fact that if $\mathcal{C}_C$ holds, then there exists a strategy σ that wins against any strategy of Perturbator, this in particular includes the strategies of Perturbator where the perturbations are chosen at random.

Let us show that if $\mathcal{C}_P$ holds then Perturbator wins almost-surely. In order to obtain the desired result, we first show that under $\mathcal{C}_P$, for any choice of strategies (σ, τ) any maximal play that is infinite and consistent with the couple (σ, τ) is eventually ε-far for n steps with $n > 0$.

Denote by $A_{k,n}$ the set of plays that are are either of length less than $k+n$ and cannot be extended or of length $k + n$ and are ε-far between steps $k + 1 \dots k + n$. Note the ε-far requirement can be written as a set of constraints over n variables, therefore we can express this set of plays by set of symbolic paths. Hence this set is measurable. Moreover, because k, n are both finite and at each step the choice of ε is independent of the current prefix of the play it follows that:

$$\forall \sigma,\ \forall \tau,\ \mathbb{P}^{\sigma,\tau}_{\mathcal{G}^{\mathsf{SG}}_{\delta}(\mathcal{A}),s}(A_{k,n}) \geq \left(\frac{\delta}{4(C+1)}\right)^n .$$

It follows that

$$\forall \sigma,\ \forall \tau,\ \mathbb{P}^{\sigma,\tau}_{\mathcal{G}^{\mathsf{SG}}_\delta(\mathcal{A}),s}(\exists k > 0,\ A_{k,n}) = 1\ .$$

We now consider the strategy τ for Perturbator given by $\mathcal{C}_P$ under which all cycles are non-aperiodic. It follows from Lemma 38 that there is no infinite run under τ, so $\mathbb{P}^{\sigma,\tau}_{\mathcal{G}^{\mathsf{SG}}_\delta(\mathcal{A}),s}(\phi \mid \exists k > 0,\ A_{k,n}) = 0$. Therefore, for any strategy σ for Controller we have:

$$\begin{aligned}
\mathbb{P}^{\sigma,\tau}_{\mathcal{G}^{\mathsf{SG}}_\delta(\mathcal{A}),s}(\phi) &= \mathbb{P}^{\sigma,\tau}_{\mathcal{G}^{\mathsf{SG}}_\delta(\mathcal{A}),s}(\phi \mid \exists k > 0,\ A_{k,n})\mathbb{P}^{\sigma,\tau}_{\mathcal{G}^{\mathsf{SG}}_\delta(\mathcal{A}),s}(\exists k > 0,\ A_{k,n}) \\
&\quad + \mathbb{P}^{\sigma,\tau}_{\mathcal{G}^{\mathsf{SG}}_\delta(\mathcal{A}),s}(\phi \mid \forall k > 0,\ \neg A_{k,n})(1 - \mathbb{P}^{\sigma,\tau}_{\mathcal{G}^{\mathsf{SG}}_\delta(\mathcal{A}),\mathcal{A}}(\exists k > 0,\ A_{k,n})) \\
&= \mathbb{P}^{\sigma,\tau}_{\mathcal{G}^{\mathsf{SG}}_\delta(\mathcal{A}),s}(\phi \mid \forall k > 0, \neg A_{k,n})(1 - \mathbb{P}^{\mathcal{G}^{\mathsf{SG}}_\delta(\mathcal{A}),\sigma,\tau}_s(\exists k > 0,\ A_{k,n})) = 0\ .
\end{aligned}$$

□

E.3 MDP semantics: Characterization

In the MDP semantics, for given $\delta > 0$, at each step, Controller picks a delay $d \geq \delta$, and an action a such that for every edge $e = (\ell, g, a, R, \ell')$ such that $\nu + d \models g$, we have $\nu + d + \varepsilon \models g$ for all $\varepsilon \in [-\delta, \delta]$, and there exists at least one such edge e. Then, a perturbation $\varepsilon \in [-\delta, \delta]$, and an edge satisfying the above conditions are chosen independently and uniformly at random. Thus in this semantics, Controller's strategy resolves all non-determinism, and defines a random process.

For a given timed Büchi automaton, denote ϕ the set of accepting runs. The almost-sure winning problem under the MDP semantics consists in deciding the following equation:

$$\exists \delta > 0,\ \exists \sigma,\ \mathbb{P}^{\sigma}_{s_0}(\phi) = 1. \tag{3}$$

We will present a decidable characterization on the region graph $\mathcal{R}(\mathcal{A})$. First we define formally what an MDP is.

Definition 39. *An MDP is a tuple (S, A, P), where S is a finite set of states, A is a finite set of actions, and P is the following mapping $P : S \times A \to \mathsf{dist}(S)$ where $\mathsf{dist}(S)$ is the set of probability distributions of S.*

We will see $\mathcal{R}(\mathcal{A})$ as an MDP by assigning uniform probability distributions to all available actions (r', a) at each state (ℓ, r). We then consider the deterministic Büchi automaton $\mathcal{B}$ of Lemma 20 which recognizes infinite paths in the region automaton containing infinitely many disjoint factors with complete FOGs. We consider the direct product of the MDP $\mathcal{R}(\mathcal{A})$ with the (deterministic) automaton $\mathcal{B}$, which is again a finite-state MDP, which we denote by $\mathcal{G}$. Let us denote by $F_{\mathcal{B}}$ the set of states of $\mathcal{G}$ whose projections to $\mathcal{B}$ are accepting. By an abuse of notation, we still denote by ϕ the set of states of $\mathcal{G}$ whose projections to $\mathcal{R}(\mathcal{A})$ are accepting for ϕ.

A run in an MDP is infinite sequence of states, a history is a finite run, and a strategy is a mapping that assigns to every history an action. Once an initial state s and a strategy σ are chosen in an MDP M, one can define a unique measure over the runs of M denoted $\mathbb{P}^{\sigma}_{M}$ (c.f. [9]).

Algorithm for almost-sure winning We are now ready to state two equivalent conditions on $\mathcal{R}(\mathcal{A})$, which we will then prove to characterize almost sure Büchi winning condition for the MDP semantics. The first condition $\mathcal{C}_1$ corresponds to $\mathcal{W}'$ in the core of the paper; however, we also need to introduce the equivalent condition $\mathcal{C}_2$ in order to carry out the proofs.

Lemma 40. *The following conditions are equivalent:*

($\mathcal{C}_1$) *there exists a finite-memory strategy σ for $\mathcal{R}(\mathcal{A})$ such that every reachable cycle in $\mathcal{R}(\mathcal{A})[\sigma]$ is aperiodic, and $\mathbb{P}^{\sigma}_{\mathcal{R}(\mathcal{A})}(\phi) = 1$.*

($\mathcal{C}_2$) *there exists a finite memory strategy σ for $\mathcal{G}$ such that Büchi condition $F_{\mathcal{B}}$ is surely satisfied, and $\mathbb{P}^{\sigma}_{\mathcal{G}}(\phi) = 1$.*

Proof. We start by showing that $\mathcal{C}_1 \implies \mathcal{C}_2$. Let σ be the strategy defined by $\mathcal{C}_1$. We denote by $\tilde{\sigma}$ the strategy of Controller in $\mathcal{G}$ defined for any finite prefix h of $\mathcal{G}$ by

$$\tilde{\sigma}(h) = \sigma(f(h)),$$

where f is the projection of h over the states of $\mathcal{R}(\mathcal{A})$. Since $\mathcal{B}$ does not restrict the moves of $\mathcal{R}(\mathcal{A})$ and $\mathbb{P}^{\sigma}_{\mathcal{R}(\mathcal{A})}(\phi) = 1$ it follows that $\mathbb{P}^{\tilde{\sigma}}_{\mathcal{G}}(\phi) = 1$.

We show that under $\tilde{\sigma}$, $F_{\mathcal{B}}$ is satisfied surely. Let π be an infinite run over $\mathcal{G}[\tilde{\sigma}]$ and let $\rho = f(\pi)$ be the projection of π over $\mathcal{R}(\mathcal{A})[\sigma]$. As in the proof of Lemma 3, because all cycles of $\mathcal{R}(\mathcal{A})[\sigma]$ are aperiodic, by Lemma 7, ρ contains infinitely many factors with complete FOGs. Furthermore, any path that contains a factor with a complete FOG has a complete FOG too. By construction of $\mathcal{B}$, it follows that $\mathcal{B}$ accepts ρ.

We show now that $\mathcal{C}_2 \implies \mathcal{C}_1$. Assume $\mathcal{C}_2$ and let σ be the strategy witnessing $\mathcal{C}_2$. We define the strategy $\hat{\sigma}$ for every finite prefix h over the states of $\mathcal{R}(\mathcal{A})$ as follows:

$$\hat{\sigma}(h) = \sigma(h'),$$

where $h' = f^{-1}(h)$, which is well defined since $\mathcal{B}$ is deterministic. Again since σ satisfies ϕ almost-surely in $\mathcal{G}$ and the fact that $\mathcal{B}$ does not restrict the actions of $\mathcal{R}(\mathcal{A})$, it follows that $\mathbb{P}^{\hat{\sigma}}_{\mathcal{R}(\mathcal{A})}(\phi) = 1$.

It remains to show that under $\hat{\sigma}$, every reachable cycle in $\mathcal{R}(\mathcal{A})[\hat{\sigma}]$ is aperiodic. Assume towards a contradiction the existence of a reachable non-aperiodic cycle in $\mathcal{R}(\mathcal{A})[\hat{\sigma}]$ and let $\rho_0\rho^{\omega}$ be a run in $\mathcal{R}(\mathcal{A})$ compatible with $\hat{\sigma}$ such that $\gamma(\rho)$ is non-aperiodic. Then, there exists an infinite run $\pi_0\pi^{\omega}$ in $\mathcal{G}$ compatible with σ such that $f(\pi_0) = \rho$ and $f(\pi) = \rho$. We show that $\pi_0\pi^{\omega}$ is not accepted by $\mathcal{B}$. In fact, if a complete FOG appears in this word, then any longer suffix has a complete FOG, which is not the case since for each $n >$, $\pi_0\pi^n$ has a non-aperiodic FOG. □

Before proving that these conditions characterize almost-sure winning in the MDP semantics, let us prove that they are decidable in exponential time, thus proving Lemma 16. For this, we prove a more general result showing how to compute winning states and strategies in MDPs given with one almost-sure and one sure Büchi objectives.

For an MDP M, and a subset of states $S' \subseteq S$, we say that S' induces a sub-MDP if the MDP M restricted to states of S' is an MDP. In this case, we

abuse of notation and say that S' is a sub-MDP. We also say that a strategy σ is compatible with a sub-MDP S' if, when started in any state of S', all actions prescribed by σ surely lead to states in S'.

Lemma 41. *Let M be any MDP with state space S, ϕ_1, ϕ_2 Büchi objectives. Let $Q \subseteq S$ be the largest sub-MDP satisfying the following two conditions:*

i) for every $q \in Q$ there exists a strategy σ_1 compatible with Q such that ϕ_1 is satisfied surely in Q,
ii) for every $q \in Q$ there exists a strategy σ_2 compatible with Q such that ϕ_2 is satisfied almost-surely in Q.

Then Q is the set of all states of M from which there exists a strategy σ surely satisfying ϕ_1, and almost-surely satisfying ϕ_2. Moreover, strategies with linear memory are sufficient, and one can compute Q, and a winning strategy in polynomial time.

Proof. Note that both strategies σ_1 and σ_2 can be assumed to be memoryless.

For any state $q \in Q$, let σ be the strategy defined as follows:

- apply σ_1 until a state in ϕ_1 is visited,
- apply σ_2 for $|S|$ steps,
- restart from scratch.

Let us show that σ satisfies ϕ_1 surely and ϕ_2 almost-surely from any state in Q. We denote by F_i the Büchi states associated with the condition ϕ_i for $i \in \{1, 2\}$. We start by showing that ϕ_1 is satisfied surely. Because i) holds it follows that from any state of $q \in Q$, any play compatible with σ_1 visits a state in $F_1 \cap Q$ in at most $|S|$ steps. Since σ switches back to σ_1 after a finite number of time it follows that a state in $F_1 \cap Q$ is visited once every $2|S|$ steps. It follows that ϕ_1 is surely satisfied.

We show that ϕ_2 is satisfied almost-surely. Denote by A the set of runs that start from a state in Q and visits a state in $F_2 \cap Q$ after the $|S|$ first steps. Thanks to ii) we know that under σ_2 we have: $\mathbb{P}^{\sigma_2}_{\mathcal{G}}(A) \geq p_{\min}^{|S|}$, where $p_{\min}$ denotes the least positive probability in M. Now since $p_{\min}^{|S|}$ is positive and since σ plays according to σ_2 for $|S|$ steps infinitely many times it follows that in almost all runs consistent with σ a state if F_2 is visited infinitely many times. Thus ϕ_2 is satisfied almost-surely. Now notice that any set of states satisfying ϕ_1 surely and ϕ_2 almost-surely satisfies i) and ii), thus Q is the set of all states in M satisfying ϕ_1 surely and ϕ_2 almost-surely.

We exhibit now a procedure that computes the largest set of states satisfying i) and ii). For any subset of states $S' \subseteq S$ inducing a sub-MDP, we write $\mathsf{Sure}(S', \phi_1)$ to denote an algorithm computing the set of sure winning states for the objective ϕ_1. We also write $\mathsf{AlmostSure}(S', \phi_2)$ to denote an algorithm computing set of almost-sure winning states for the objective ϕ_2. Note that both $\mathsf{Sure}(\cdot, \phi_1)$ and $\mathsf{AlmostSure}(\cdot, \phi_2)$ always return sets of states that induce sub-MDPs. We define $W_1 = S$, and $W_i = \mathsf{AlmostSure}(\mathsf{Sure}(W_{i-1}, \phi_1), \phi_2)$. Define W^* as the fixpoint of this sequence.

Let us show that from any state in W^* there exists a strategy σ surely satisfying ϕ_1, and almost-surely satisfying ϕ_2. Notice that by definition of

$\mathsf{AlmostSure}(\cdot, \phi_2)$ and $\mathsf{Sure}(\cdot, \phi_2)$ we know that $\mathsf{AlmostSure}(S', \phi_2) \subseteq S'$ and $\mathsf{Sure}(S', \phi_2) \subseteq S'$ for any sub-MDP S'. It follows that $\mathsf{AlmostSure}(W^*, \phi_2) = W^*$ and $\mathsf{Sure}(W^*, \phi_1) = W^*$, thus W^* satisfies $i)$ and $ii)$ and from the first part of the proof we obtain that all states in W^* satisfy ϕ_1 surely and ϕ_2 almost-surely. This shows the correction of our procedure.

We now need to prove that W^* contains all states from which there exists a strategy σ surely satisfying ϕ_1, and almost-surely satisfying ϕ_2. We show that any state outside W^* is either not surely winning with respect to ϕ_1 or not almost-surely winning with respect to ϕ_2.

We prove by induction on $i > 0$ that all states outside W_i are losing. This is trivial for $i = 1$ since $W_1 = S$. Let s be any state not in W^*, and let $1 < i = \min\{j > 0 \mid s \notin W_j\}$. We can write then that $s \in W_{i-1}$ and $s \notin W_i$, this implies that either $a)$: $s \notin \mathsf{Sure}(W_{i-1}, \phi_1)$ or that $b)$: $(s \in \mathsf{Sure}(W_{i-1}, \phi_1)) \wedge (s \notin \mathsf{AlmostSure}(W_{i-1}, \phi_2))$. If $a)$ holds then for any strategy σ, either σ is not compatible with W_i when started from s, which means it leaves W_i, and it is losing by induction; or σ is compatible with W_i and there exists an infinite run consistent with σ in the sub-MDP W_{i-1} and starting at s that is not surely winning with respect to ϕ_1. If $b)$ holds, then any strategy σ compatible with the sub-MDP $\mathsf{Sure}(W_{i-1}, \phi_1)$ is not almost-surely winning with respect to ϕ_2. Since strategies that are not compatible with $\mathsf{Sure}(W_{i-1}, \phi_1)$ are losing as we showed in the case $a)$, we get that no strategy from s satisfies ϕ_1 surely and ϕ_2 almost-surely.

Hence, W^* is exactly the set of states from which some strategy satisfies ϕ_1 surely and ϕ_2 almost surely.

Last, W^* can be computed in polynomial time since both $\mathsf{AlmostSure}(\cdot, \phi_2)$ and $\mathsf{Sure}(\cdot, \phi_2)$ can be computed in polynomial time and at each iteration we have $|W_i| < |W_{i-1}|$. □

It follows that since $\mathcal{R}(\mathcal{A})$ has exponential size, the conditions of Lemma 40 can be decided in exponential time, and exponential-memory strategies exists in $(\mathcal{R}(\mathcal{A}), \mathcal{W}')$, proving Lemma 16.

Proof of Correctness It remains to prove that these conditions are equivalent with almost-sure winning in the MDP semantics, proving Proposition 15.

We first prove Lemma 42 which allows one to lift a strategy in the MDP semantics to $\mathcal{R}(\mathcal{A})$.

Lemma 42. *Let σ be any strategy in $\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A})$. Let X be a measurable subset of $\mathsf{Runs}(\mathcal{A}, s)$ with positive probability. There exists a strategy $\hat{\sigma}$ in $\mathcal{R}(\mathcal{A})$ such that for any path h of $\mathcal{R}(\mathcal{A})$, $\mathbb{P}^{\hat{\sigma}}_{\mathcal{R}(\mathcal{A}), v_0}(h) = \mathbb{P}^{\sigma}_{\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A}), s_0}(\mathsf{proj}^{-1}(h) \mid X)$, where v_0 (resp. s_0) is the initial state of $\mathcal{R}(\mathcal{A})$ (resp. $\mathcal{A}$).*

Proof (of Lemma 42). For any history h, we define the strategy $\hat{\sigma}$ by

$$\hat{\sigma}((r, a) \mid h) = \mathbb{P}^{\sigma}_{\mathcal{A}, s_0}(\mathsf{proj}^{-1}(h(r, a)) \mid \mathsf{proj}^{-1}(h) \cap A),$$

if $\mathbb{P}^{\sigma}_{\mathcal{A}, s_0}(h \cap A) > 0$ and arbitrarily otherwise. We show the required property by induction $n \geq 1$, the length of h.

The base case $n = 1$ is trivial since $\mathbb{P}^{\hat{\sigma}}_{\mathcal{R}(\mathcal{A}), r_0}(r) = \mathbb{P}^{\sigma}_{\mathcal{A}, s_0}(\mathsf{proj}^{-1}(r) \mid A)$, where the value is 1 if $r = r_0$ and 0 otherwise.

Consider the property holds up to length n. Consider a path $h(r,a)$ of $\mathcal{R}(\mathcal{A})$ of length $n+1$. We write

$$\begin{aligned}\mathbb{P}^{\hat{\sigma}}_{\mathcal{R}(\mathcal{A}),r_0}(h(r,a)) &= \mathbb{P}^{\hat{\sigma}}_{\mathcal{R}(\mathcal{A}),r_0}(h)\cdot\hat{\sigma}((r,a)\mid h)\\ &= \mathbb{P}^{\sigma}_{\mathcal{A},s_0}(\mathsf{proj}^{-1}(h)\mid A)\cdot\hat{\sigma}((r,a)\mid h)\end{aligned}$$

Here the second line is by induction. Now, if $\mathbb{P}^{\sigma}_{\mathcal{A},s_0}(\mathsf{proj}^{-1}(h)\cap A)=0$, then we also have $\mathbb{P}^{\sigma}_{\mathcal{A},s_0}(\mathsf{proj}^{-1}(h)\mid A)=0$. It follows that $\mathbb{P}^{\hat{\sigma}}_{\mathcal{R}(\mathcal{A}),r_0}(h(r,a))=0$ but also $\mathbb{P}^{\sigma}_{\mathcal{A},s_0}(\mathsf{proj}^{-1}(h(r,a)))=0$, as required. Assuming $\mathbb{P}^{\sigma}_{\mathcal{A},s_0}(\mathsf{proj}^{-1}(h)\cap A)>0$, we get

$$\begin{aligned}\mathbb{P}^{\hat{\sigma}}_{\mathcal{R}(\mathcal{A}),r_0}(h(r,a)) &= \mathbb{P}^{\sigma}_{\mathcal{A},s_0}(\mathsf{proj}^{-1}(h)\mid A)\cdot\mathbb{P}^{\sigma}_{\mathcal{A},s_0}(\mathsf{proj}^{-1}(h(r,a))\mid \mathsf{proj}^{-1}(h)\cap A)\\ &= \mathbb{P}^{\sigma}_{\mathcal{A},s_0}(\mathsf{proj}^{-1}(h(r,a))\mid A),\end{aligned}$$

which concludes the proof. □

Last, we need to construct a set of paths with nonzero measure which contain infinitely many ε-far factors. We will then use ideas similar to the stochastic game semantics case, once we condition on this event.

Lemma 43. *There exists a subset Z of $\mathsf{Runs}(\mathcal{A},s_0)$ and some $\eta>0$ such that under any strategy σ, $\mathbb{P}^{\sigma}_{\mathcal{G}^{\mathsf{MDP}}_{\delta}(\mathcal{A}),s_0}(Z)\geq\eta$. In addition, for every integer N_1, there exists $N_2\geq N_1$ such that for any run $\rho\in Z$, the run is ε-far during N consecutive steps between positions N_1 and N_2.*

Proof. Assuming the timed automaton $\mathcal{A}$ has C clocks, we let $\varepsilon=\frac{\delta}{4(C+1)}$. In any configuration (ℓ,ν) reached when playing according to $\mathsf{Controller}$'s strategy σ, there exists an interval $I\subseteq[-\delta,\delta]$ such that $|I|\geq\varepsilon$, and any perturbation d' picked in I is such that the resulting valuation $\nu+d'$ is ε-far. As a consequence, with probability at least $\frac{|I|}{2\delta}\geq\frac{1}{8(C+1)}>0$, $\mathsf{Perturbator}$ picks a delay resulting in an ε-far valuation. Observe also that the selections of perturbations constitute independent events.

We recall the following standard mathematical property on infinite products.

Lemma 44. *Let us fix some real number α in the open interval $(0,1)$. We consider the infinite product $\prod_{j=1}^{\infty}(1-\alpha^j)$ which is known to be convergent to some number $f(\alpha)$ in the open interval $(0,1)$.*

Let us fix some positive natural integer N. Given an integer $j\geq 1$, we consider the following event X_j: the perturbations selected by $\mathsf{Perturbator}$ at steps $j*N+1, j*N+2,\ldots,(j+1)*N$ are such that the resulting valuations are ε-far. By the above observations, the event X_j has probability at least $(\frac{1}{8(C+1)})^N$ which is strictly positive, and the events $(X_j)_j$ are iid.

We let $\eta=(\frac{1}{8(C+1)})^N$.

We define an increasing sequence of natural numbers $(n_k)_{k\in\mathbb{N}}$ as follows

$$n_0=0 \qquad n_k=n_{k-1}+k$$

For all $k\in\mathbb{N}_{>0}$, we define the event $Y_k=\bigvee_{j=n_{k-1}+1}^{n_k}X_j$. Observe that as $(X_j)_j$ are iid, we have that $(Y_k)_k$ are independent events as the intervals of

natural numbers $[n_{k-1}+1, n_k]$ are disjoint for distinct values of k. The following computations give:

$$\begin{aligned}
\mathbb{P}^\sigma(Y_k) &= 1 - \mathbb{P}^\sigma(\neg Y_k) \\
&= 1 - \mathbb{P}^\sigma(\textstyle\bigwedge_{j=n_{k-1}+1}^{n_k} \neg X_j) \\
&= 1 - \textstyle\prod_{j=n_{k-1}+1}^{n_k} \mathbb{P}^\sigma(\neg X_j) \\
&= 1 - \textstyle\prod_{j=n_{k-1}+1}^{n_k} (1 - \mathbb{P}^\sigma(X_j)) \\
&= 1 - (1 - \mathbb{P}^\sigma(X_1))^k \\
&\geq 1 - (1-\eta)^k
\end{aligned}$$

Then, we consider the event $Z = \bigwedge_{k \geq 1} Y_k$. As events $(Y_k)_k$ are independent, we can easily prove, using Lemma 44:

$$\mathbb{P}^\sigma(Z) = \prod_{k=1}^{\infty} \mathbb{P}^\sigma(Y_k) \geq \prod_{k=1}^{\infty} (1 - (1-\eta)^k) = f(1-\eta) > 0$$

□

Proof (of Proposition 15). We first suppose that Controller wins the abstract game $(\mathcal{R}(\mathcal{A}), \mathcal{W}')$. By Lemma 16, there exists a finite-memory winning strategy $\hat{\sigma}$. By definition of $\mathcal{W}'$, every outcome of $\mathcal{R}(\mathcal{A})[\hat{\sigma}]$ contains infinitely many factors whose FOG is complete. Because condition $\mathcal{C}_1$ is satisfied, this entails that every reachable cycle in $\mathcal{R}(\mathcal{A})[\hat{\sigma}]$ has an aperiodic FOG. Then, we apply the results of Section 5 to derive a positive value of δ and a concrete strategy σ which surely ensures infinite runs, that is, against any choice of the perturbations and the edges (in fact, condition $\mathcal{C}_C$ holds if we consider a trivial Büchi condition). However, we still need to show that σ satisfies ϕ almost surely. By construction, the strategy σ ensures that for any path h of $\mathcal{R}(\mathcal{A})$ compatible with $\hat{\sigma}$, all runs of $\mathcal{G}_\delta(\mathcal{A})$ compatible with σ are in $\mathsf{proj}^{-1}(h)$, regardless whether the perturbations are chosen randomly or adversarially. It follows that, in the MDP semantics, for any path h, $\mathbb{P}^\sigma_{\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A}), s_0}(\mathsf{proj}^{-1}(h)) = \mathbb{P}^\sigma_{\mathcal{R}(\mathcal{A}), v_0}(h)$. Now, because $\hat{\sigma}$ almost surely satisfies ϕ in $\mathcal{R}(\mathcal{A})$, strategy σ almost surely satisfies ϕ in $\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A})$.

Conversely, suppose that Controller loses the game $(\mathcal{R}(\mathcal{A}), \mathcal{W}')$. We proceed by contradiction, and suppose that there exists a winning strategy σ in the concrete game $\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A})$. Using Lemmas 42 and 43, we build a strategy $\hat{\sigma}$ from σ and Z. The property ensured by Lemma 42 and the fact that σ satisfies ϕ almost-surely allows to show that $\hat{\sigma}$ also does (in $\mathcal{R}(\mathcal{A})$). As Controller loses the game $(\mathcal{R}(\mathcal{A}), \mathcal{W}')$, this entails that there exist an infinite execution w in $\mathcal{R}(\mathcal{A})[\sigma]$ and a bound N_0 after which no factor of w has a complete FOG. Let N' be the integer given by Lemma 43 and N_0, and consider the prefix $w_{N'}$ of w of length N'. As Z has positive measure and by Lemma 42, we have that $\mathbb{P}^\sigma_{\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A}), s_0}(\mathsf{proj}^{-1}(w_{N'}) \cap Z) > 0$. Moreover, as shown in Section 5, any run ε-far during N consecutive steps along non-aperiodic cycles is necessarily blocking. Thus, with positive probability, strategy σ is losing, contradiction. □

Algorithm for limit-sure winning We now show that the characterization for the limit-sure winning is correct, namely: Controller wins limit-surely in the MDP semantics iff Controller can satisfy $\mathcal{W}''$ in $\mathcal{R}(\mathcal{A})$.

First notice that for any MDP, the set of states satisfying a reachability objective almost-surely can be computed in polynomial time. Thus, Lemma 18 follows.

The proof of Proposition 17 uses the notion of end component. An *end-component* of an MDP is a strongly connected sub-MDP (see [6]). It is well-known that there are finitely many end-components, and that almost-surely, the set of states and actions seen infinitely often defines an end-component. Because the union of two end-components with non-empty intersection is again an end-component, one can define *maximal end-components (MECs)*.

For any MDP M, and MEC D, we say that a run ρ *enters* D, if there exists $i > 0$ such that $\rho_i \in D$. We say that ρ *leaves* D, if there exists $j > i$ such that $\rho_j \notin D$. A MEC D is *bottom* if no other MEC D' is reachable from D. It is easy to see that in all MDPs, there exists at least one bottom MEC, and for all states s, and strategies σ, runs that leave all non-bottom MECs they enter eventually enter a bottom MEC almost surely.

Proof (of Proposition 17). We first show that if Controller satisfies $\mathcal{W}''$ in $\mathcal{R}(\mathcal{A})$ from r_0 then he wins in the MDP semantics limit-surely. Suppose that Controller wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W}'')$. Then there exists a memoryless strategy $\hat{\sigma}_1$ such that any play compatible with σ in $\mathcal{R}(\mathcal{A})$ that starts is r_0 reaches almost-surely the set Win$'$, and a finite-memory strategy $\hat{\sigma}_2$ satisfying $\mathcal{W}'$ from states in Win$'$.

Let $\varepsilon > 0$. There exists an integer N such that at step N, with probability at least $1 - \varepsilon$, the N-th step of any play consistent with $\hat{\sigma}_1$ and starting in r_0 is in the set Win$'$. Consider the unfolding of $\mathcal{R}(\mathcal{A})[\hat{\sigma}_1]$ from the initial vertex of $\mathcal{R}(\mathcal{A})$, and of depth N. As we are interested in paths of finite length, one can compute shrinking matrices and an upper bound on δ such that the resulting shrunk zones are non-empty (see Lemma 10). We can thus derive a concrete strategy σ_1 along this finite unfolding, which preserves the probabilities of $\hat{\sigma}_1$. In particular, with probability at least $1 - \varepsilon$, the runs compatible with σ_1 have reached the set Win$'$ after at most N steps. Using the construction done for the almost-sure case, we know that there exists an upper bound on δ and a concrete strategy σ_2 which wins almost-surely the game $\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A})$ w.r.t. the Büchi objective from states in Win$'$. This yields the desired implication.

Let us assume that Controller wins in the MDP semantics limit-surely, and prove that he satisfies $\mathcal{W}''$ in $\mathcal{R}(\mathcal{A})$ from r_0. By contradiction, suppose that it is not the case. Consider the set of states of the MDP $\mathcal{R}(\mathcal{A})$ from which there is no path to the set Win$'$, and denote this set Lose$'$ (observe that this set is absorbing: once entered, one can not leave it). Because $\mathcal{R}(\mathcal{A})$ is a finite MDP, and $\mathcal{W}''$ is not satisfied almost surely from v_0, there exists $\nu > 0$ such that for every strategy $\hat{\sigma}$, the set Lose$'$ is reached with probability at least ν from v_0.

Consider $\varepsilon < \nu$, and fix some $\delta > 0$ and some strategy σ for $\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A})$ such that $\mathbb{P}^\sigma_{\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A}), s_0}(\phi) \geq 1 - \varepsilon$, which exist by assumption. Let $\hat{\sigma}$ given by Lemma 42. By the above remark, it follows that $\mathbb{P}^\sigma_{\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A}), v_0}(\mathsf{proj}^{-1}(\text{F Lose'})) \geq \nu$ where the property F Lose' means that a play compatible with $\hat{\sigma}$ eventually enters the set Lose'.

We will show that the set Lose' is almost-surely losing w.r.t. objective ϕ (recall that ϕ is a Büchi objective with accepting states B) in the MDP semantics, which yields to a contradiction.

Let us distinguish the end-components inside Lose'. We will define events which lead to losing unless the end-component is left. For any MEC e,

1. if $e \cap B = \emptyset$, then any run $\rho \in \mathsf{proj}^{-1}(\text{F Lose'})$ which enters e, if ρ stays forever inside e, then ρ is losing w.r.t. objective ϕ. Let us call the set of such runs L_e. Thus, when a run enters e, either L_e occurs, or e is eventually left.
2. If $e \cap B \neq \emptyset$, then we define a set of runs Z_e which are losing. We recall that the second objective in the condition $\mathcal{W}'$ (abstract MDP in $\mathcal{R}(\mathcal{A})$) is to ensure that every outcome contains infinitely many disjoint factors whose FOG is complete. This can be understood as a Büchi objective in the product MDP $\mathcal{R}(\mathcal{A}) \parallel \mathcal{B}$ (the deterministic Büchi automaton $\mathcal{B}$ has been defined before, see Lemma 20). We denote this objective by the set of states B'.
   Let us denote by Lose(GF B') the set of states in $\mathcal{R}(\mathcal{A})$ that are losing for Controller in the two-player game ($\mathcal{R}(\mathcal{A})$ is now viewed as two-player game and not as an MDP) restricted to e w.r.t. objective $GF\ B'$. Runs that stay infinitely in e must visit infinitely often states of $e \cap \text{Lose(GF B')} \neq \emptyset$ since otherwise we can show that $e \subseteq \text{Win}'$ (combine $\hat{\sigma}$ with a strategy winning the objective GF B', see Lemma 41).
   Let q be such a state. As the above objective is considered in a two-player finite-state (Büchi) game, q is losing implies that Perturbator has a finite-memory strategy τ to ensure Controller is losing, *i.e.* a finite lasso whose cycle contains no state in B'. In addition, the memory needed is bounded in terms of the size of $\mathcal{R}(\mathcal{A}) \parallel \mathcal{B}$. We can show as we did before that the cycle reached is actually non-aperiodic. We can thus exhibit an integer N_f such that the following this lasso during N_f steps is impossible if in addition the valuations are always α-far, with $\alpha = \frac{\delta}{4(C+1)}$.
   Consider now runs in the MDP semantics compatible with σ which belong to $\mathsf{proj}^{-1}(\text{F Lose'})$. Now, when some state in $\mathsf{proj}^{-1}(q)$ is reached, unless e is left before N_f steps, with probability at least p^{N_f}, the (stochastic) resolutions of non-determinism are compatible with τ, and the run is α-far for N_f steps.[8]
   Let us call Z_e the set of such runs. We saw earlier that all runs in Z_e are losing (c.f. proof of Proposition 15). Since states of Lose(GF B') are visited infinitely often, it follows that when any run enters e, either Z_e occurs or e is eventually left.

It follows from the above case by case analysis that non-losing runs almost surely reach bottom end-components. But because a bottom end-component cannot be left, such runs are also losing.

This shows that any run entering Lose' is almost surely losing, as required and yield the completeness of the characterization. □

References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[8] One can choose $p = \frac{1}{d8(C+1)}$ where d is the maximal degree of non-determinism. Indeed, with probability at least $\frac{1}{d}$ the resolution of non-determinism is compatible with τ, and with probability at least $\frac{1}{8(C+1)}$, the selection of the perturbation yields an α-far valuation.

2. Christel Baier, Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Marcus Größer. Almost-sure model checking of infinite paths in one-clock timed automata. In *Proceedings of the 23rd Annual IEEE Symposium on Logic in Computer Science (LICS'08)*, p. 217–226, Pittsburgh, Pennsylvania, USA, June 2008. IEEE Computer Society Press.
3. Christel Baier, Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Marcus Größer. A probabilistic semantics for timed automata. Research Report LSV-08-13, Laboratoire Spécification et Vérification, ENS Cachan, France, April 2008. 42 pages.
4. Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, LNCS 2098, p. 87–124. Springer, 2004.
5. Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robust reachability in timed automata: A game-based approach. In ICALP'12, LNCS 7392, p. 128–140. Springer, 2012.
6. Luca de Alfaro. *Formal verification of probabilistic systems.* Ph.d. thesis, Stanford University, 1997.
7. C.H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.
8. Anuj Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10(1-2):87–113, 2000.
9. Martin L. Putterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley and Sons, New York, NY, 1994.
10. Ocan Sankur, Patricia Bouyer, and Nicolas Markey. Shrinking timed automata. In FSTTCS'11, LIPIcs 13, p. 375–386. Leibniz-Zentrum für Informatik, 2011.
11. Ocan Sankur, Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. Robust controller synthesis in timed automata. In CONCUR'13, LNCS 8052, p. 546–560. Springer, 2013.